Inventing Requirements from Software: An Empirical Investigation with Web Services

Konstantinos Zachos & Neil Maiden

City University London, Centre for HCI Design
London EC1V 0HB, UK
k.zachos@soi.city.ac.uk, n.a.m.maiden@city.ac.uk

Abstract

Service-centric software systems offer new opportunities for requirements processes. This paper reports a new tool designed to increase the completeness of system requirements using information about designs and implementations of web services. It presents an algorithm for retrieving web services in domains that are analogical to a current requirements problem, to support creative thinking about requirements for that problem. It describes how the algorithm parses and analogically matches natural language descriptions of system requirements and web service descriptions. The paper also reports 2 evaluations of the tool that demonstrate improvements to specifications of requirements for a system in the automotive domain.

1. Requirements Engineering with Web Services

Service-centric systems discover, compose, invoke and monitor web services – software operations published by third-party providers independent of where the service is executed [19]. We conjecture that the increasing availability of web services from third-party providers can change requirements processes for service-centric systems. Stakeholders and analysts can retrieve relevant web services early in the process then discover new requirements by reviewing and working backwards from designs and implementations of these services based on what might be possible.

In the EU-funded SeCSE Integrated Project we have researched new tools and techniques to increase requirements completeness from retrieved web services. Evaluations with industrial partners already revealed that reviewing retrieved services can lead analysts to specify previously undiscovered requirements that they ranked as more novel compared with requirements discovered with established use case walkthrough [21]. In this paper we report new results from the next phase, in which we investigated whether retrieving web service designs and implementations from domains analogical to the current problem could support effective creative thinking about requirements.

Requirements engineering is a creative process in which stakeholders and analysts work together to create ideas expressed as system requirements [8]. However stakeholders on their own struggle to create requirements because most lack the knowledge of possible design spaces necessary to specify system requirements [12]. Therefore Robertson argues [12], analysts need to explore design spaces to invent system requirements with stakeholders.

Previously we ran creativity workshops in which stakeholders collaborated with analysts and designers to invent requirements using creativity techniques. Although successful in terms of the numbers and impact of the requirements generated [8, 9], workshops involved up to 20 stakeholders, analysts and designers for 2 days. Therefore alternative, more accessible sources of design knowledge for stakeholders and analysts were sought.

One such source is public registries of web services. Because web software services are accessed via the Internet, analysts can access and exploit them directly. And as service-centric computing grows the volume and range of available web services will increase [4], thus providing new and potentially large sources of design knowledge to be exploited. However, how can we exploit these web services? We have already shown that analysts can use web service designs and implementations to discover new and more novel requirements within an automotive domain [21]. In this paper we report new results that reveal that analysts can invent requirements from web services across domains for which these services were implemented through analogical reasoning. We know that analysts can reason analogically about requirements [7], but can it happen with analogical web services?

In section 2 we report AnTiQue (Analogy Tracker in Service Queries), a new software module to retrieve web services in domains analogical to a problem. We developed AnTiQue to answer 2 research questions:

Q1: Can AnTiQue automatically retrieve web services from domains analogical to a specified problem?

Q2: Can analysts reason with analogical services retrieved by AnTiQue to invent previously unspecified requirements ranked as more novel?

Section 3 describes AnTiQue. Sections 4 and 5 report results from a multi-phase evaluation study that provided data with which to answer the 2 questions. Section 6 uses this data to answer the 2 research questions. The paper ends with future research directions.

2. SeCSE's Requirements Process and Service Discovery Environment

SeCSE supports an iterative requirements process for service-centric systems [2]. Analysts form service queries from requirements specifications to retrieve web services compliant with the requirements. Descriptions of retrieved services are presented to analysts who use them to refine and complete requirements to enable more accurate service retrieval, and so on. Analysts rarely express requirements at the correct levels of abstraction and granularity to retrieve all relevant web services immediately, so relevance feedback from retrieved services also enables analysts to specify new requirements and re-express current ones to increase the likelihood of discovering new web services.

To ensure industrial uptake SeCSE's requirements process uses established techniques based on structured natural language. Analysts specify service-centric system behaviour with UML use case specifications and required system properties in a testable form with VOLERE shells [13]. The process extends the Rational Unified Process (RUP) without mandating additional specification or service retrieval activities [22].

To support SeCSE's requirements process we implemented the SeCSE service discovery environment. The original environment had 3 modules: (i) service registries; (ii) UCaRE, a module to describe requirements and generate service queries, and; (iii) EDDiE, the service discovery engine. To provide new support for requirements invention we replaced one of these modules – EDDiE – with a new one called AnTiQue. AnTiQue retrieves web services from domains that are analogical to the current domain.

2.1 The Service Registries

The environment discovers web services from registries that link to service implementations that applications invoke and facets that specify different aspects of services. Current registries such as UDDI are inadequate for retrieving services using criteria such as quality of service and exception handling. Therefore SeCSE has defined 7 facets of a service including signature, description and quality-of-service [15] that describe information about web services using XML data structures. Service discovery in SeCSE uses the description and quality-of-service facets to retrieve web services. Figure 1 shows part of the service description facet of one web service from the reported evaluations. SeCSE's service registries are implemented using eXist, an Open Source native XML database featuring index-based XQuery processing, automatic indexing.

> A tourist in London wants to find the nearest underground station. The tourist uses his mobile phone to find the location of the nearest stations. The tourist uses an application to request the names of the nearest stations. The application retrieves the names of the 3 nearest stations, and their distances from the tourist's location.

Figure 1. Part of the specification of the *Find Nearby Station* web service from the evaluation

2.2 The UCaRE Requirement Component

Analysts express requirements for new applications using UCaRE, a web-based .NET application depicted in Figure 2. UCaRE supports tight integration of use case and requirements specifications – a requirement expressed using VOLERE can describe a system-wide requirement, a requirement on the behavior specified in one use case, or a requirement on behavior expressed in one use case action.

An analyst manages requirements and use cases through a web client. UCaRE allows analysts to create service queries from use case and requirements specifications. At the start of the requirements process analysts work with stakeholders to develop simple use case précis that describe the required behaviour of the service-centric system. Figure 3 shows a typical précis in UCaRE, defining what a driver might want from an in-car car parking booking system. In the second stage, the analyst selects elements of the specification to include in a service query.

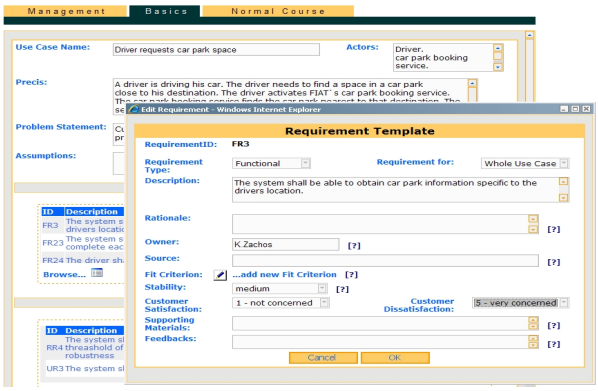

Figure 2. Example use case and requirement specification specified in UCaRE

> A driver is driving his car. The driver needs to find a space in a car park close to his destination. The driver activates FIAT`s car park booking service. The car park booking service finds the car park nearest to that destination. The service will check if there is a space in that car park, and if so it books the space.

Figure 3. The use case précis for the car park booking system, which is used to formulate queries with which to discover services

In the original environment analysts can manipulate

specified use cases and requirements to generate service queries that are fired at service registries with EDDiE to retrieve web services from the same domain as the current problem. This service discovery engine [22] implements advanced term disambiguation and query expansion algorithms to add different terms with similar meanings to the query using the WordNet online lexicon, thus increasing the number of web services retrieved from the registries. Analysts can then reject retrieved web services prior to specifying new requirements from the retained ones.

To support cross-domain analogical invention of requirements we replaced EDDiE with AnTiQue, a new module for analogical service discovery.

3. The AnTiQue Module

The purpose of AnTiQue is to retrieve designs and implementations of web services that service providers designed for domains that are analogical to the current requirement problem. AnTiQue's design seeks to solve 2 research problems: (i) match incomplete and ambiguous natural language descriptions of requirements and web services from different parties using different lexical terms; (ii) compute complex analogical matches between descriptions without a priori classification of the described domains.

For example, car drivers use a service-centric system to locate and book parking spaces at their destinations. We have already shown that analysts can use SeCSE's Service Discovery Environment to retrieve and use design information about retrieved web services in the same domain – car parking – to inform requirements specification [21]. Analogical service retrieval can increase the number of web services that are useful to the requirements process by retrieving services from other domains, for example services that *find and book cinema tickets*, *locate and reserve hotel rooms*, and *select and reserve places at a summer school*. The design and implementation of each web service might have features that, through analogical reasoning, can trigger discovery of new requirements on the car park booking system. For example, just as a *hotel reservation system* allows customers to *book rooms of different sizes*, an analogical requirement is to allow the driver to *reserve different sizes of parking spaces for different vehicle sizes*. AnTiQue seeks to leverage these new sources of design knowledge in a requirements process.

Analogical retrieval in AnTiQue uses a similarity model called the Structure Mapping Theory (SMT) [1], which seeks to transfer a network of related facts rather than unrelated one [1] from a source (a web service) to a target domain (the requirements problem). AnTiQue's implementation of the SMT parses and represents natural language statements from use case and requirement-based service queries as predicates in the form of prepositional networks of nodes (objects) and edges (predicate values). It represents 2 kinds of predicate. *Attributional* predicates state properties of objects in the form *PredicateValue(Object)*. *Relational* predicates express relations between objects in the form *PredicateValue(Object1, Object2)*. For instance *the car is red* becomes *red(car)* and *the driver drives the car* becomes *drive(driver, car)*. According to the SMT an *analogy* is a comparison in which relational predicates, but few or no attributional predicates, can be mapped from a source to a target.

For example analogical inferences about *reserving a car park space* from a mapping with *booking a cinema ticket* concern the shared relational structures, in that *a customer books a cinema ticket* (*book(customer, cinema ticket)*), just as *a driver books a car park space* (*book(driver, car park space)*) but not the attribute similarities. On the other hand, a *literal similarity* statement is a comparison in which a large number of attributional and relational predicates are mapped from a source to a target. For example the attributional predicates *customer(person)* and *driver(person)* indicate some level of literal similarity.

Figure 4 depicts AnTiQue's 5 components. In the first a service query generated by an analyst is divided into sentences, then part-of-speech tagged, shallow parsed to identify sentence constituents (noun groups, verbs…) and chunked in noun phrases. In the second the algorithm applies a set of rules and heuristics to identify predicates in each sentence structure. Natural language sentences are presented as predicates in the form *PredicateValue(Object1, Object2)*. In the third the algorithm expands each predicate with additional predicate values that have similar meaning according to verb classes found in VerbNet to increase the likelihood of a match with a web service description. For example the predicate value *find* (taken from the predicate *find(x,y)*) is in the same verb class as *locate* which is also included in the predicate list (as *locate(x,y)*). The fourth component matches all expanded predicates to a similar set of predicates (pre-processed using the first 2 components) that describe each candidate web service from the service description facet in the SeCSE service registry. It uses XQuery text-searching functions to discover an initial set of web service descriptions that satisfy global search constraints. The fifth component applies semantic and dependency-based similarity measures to refine the candidate service set. AnTiQue returns an ordered set of analogical services based on the match score with the service query.

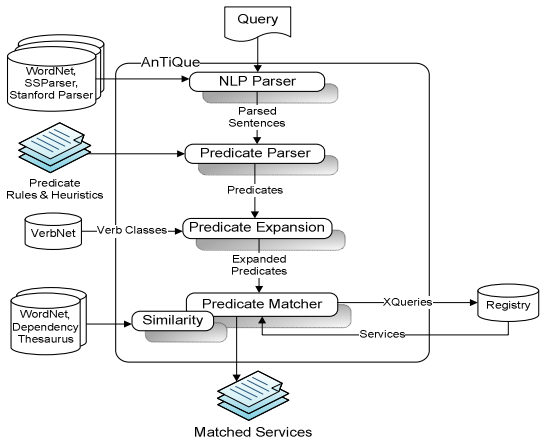

Figure 4. Internal structure of AnTiQue

The components use WordNet, VerbNet, and the Dependency Thesaurus to compute attributional and relational similarities. WordNet is a lexical database inspired by psycholinguistic theories of human lexical memory [20]. Its word senses and definitions provide the data with which to disambiguate terms in SeCSE service queries. Its semantic relations link terms to other terms with similar meanings with which to make service queries more complete. For example a service query with the term *car* is expanded with other terms with similar meaning, such as *automobile* and *vehicle*, to increase matches with web service descriptions.

VerbNet [3] is a domain independent verb lexicon. It organizes terms into verb classes that refine Levin [5] classes and add sub-classes to achieve syntactic and semantic coherence among members of a verb class. AnTiQue uses it to expand service query predicate values with different members from the same verb class. For example, service queries with the verb *book* are expanded with other verbs with similar meaning such as *reserve* and *order*.

The Dependency Thesaurus supports dependency-based word similarity matching to detect similar words from text corpora. Lin [6] used a 64-million word corpus to compute pair-wise similarities between all of the nouns, verbs, adjectives and adverbs in the corpus using a similarity measure. Given an input word the Dependency Thesaurus can retrieve similar words and group them automatically into clusters. AnTiQue used the Dependency Thesaurus to compute the relational similarity between 2 sets of predicates.

In the remainder of this section we demonstrate the AnTiQue components using text from the example web service and use case descriptions in Figures 1 and 3.

3.1 The Natural Language Processing Parser

This component prepares the structured natural language (NL) service query for predicate parsing and expansion. In the first step the text is split into sentences. In the second a part-of-speech tagging process is applied that marks up the words in each sentence as corresponding to a particular lexical category (part-of-speech) using its definition and context. In the third step the algorithm applies a NL processing technique called *shallow parsing* that attempts to provide some machine understanding of the structure of a sentence without parsing it fully into a parsed tree form. The output is a division of the text's sentences into a series of words that, together, constitute a grammatical unit. In our example the tagged sentence *the driver needs to find a space in a car park close to his destination* is shown in Figure 5. Tags that follow a word with a forward slash (e.g. *driver/NN*) correspond to lexical categories including noun, verb, adjective and adverb. For example, the *NN* tag means "noun singular or mass", *DT* means "determinant" and *VBZ* means "verb, present tense, 3rd person singular". Tags attached to each chunk (e.g. *[The/DT driver/NN]*$_{NP}$) correspond to phrasal categories. For instance, the *NP* tag denotes a "noun phrase", *VP* a "verb phrase", *S* a "simple declarative clause", *PP* a "prepositional phrase" and *ADVP* a "adverb phrase".

```
[The/DT driver/NN]NP [needs/VBZ]VP [to/TO]S [find/VB]VP [a/DT
space/NN]NP [in/IN]PP [a/DT car\_park/NN]NP [close/RB]ADVP [to/TO]PP
[his/PRP$ destination/NN] NP.
```

Figure 5. The sentence *the driver needs to find a space in a car park close to his destination* after performing part-of-speech tagging and chunking

The component then decomposes each sentence into its *phrasal categories* used in the next component to identify predicates in each sentence structure.

3.2 The Predicate Parser

This component automatically identifies predicate structures within each annotated NL sentence based on *syntax structure rules* and *lexical extraction heuristics*. Syntax structure rules break down a pre-processed NL sentence into sequences of phrasal categories where each sequence contains 2 or more phrasal categories. Lexical extraction heuristics are applied on each identified sequence of phrasal categories to extract its lexical content used to generate one or more predicates.

Firstly the algorithm applies 21 syntax structure rules. Each rule consists of a phrasal category sequence of the form $R_i \rightarrow [B_j]$, meaning that the rule R_i consists of a phrasal category sequence $B_1, B_2, \ldots, B_j$. For example the rule $R_4 \rightarrow [NP, VP, S, VP, NP]$ reads: rule R_1 consists of a *NP* followed by a *VP*, a *S*, a *VP*, and a *NP*, where *NP*, *VP* and *S* mean a *noun phrase*, a *verb phrase* and a *simple declarative clause* respectively. The method takes a phrasal category list as input and returns a list containing each discovered syntax struc-

ture rule and its starting point in the corresponding phrasal category list, e.g. *{(R1,3), (R5,1)}*. In our example, the input for the pre-processed sentence shown in Figure 5 corresponds to a list *Input = (NP, VP, S, VP, NP, PP, NP, ADVP, PP, NP)*. Starting from the first list position the method recursively checks whether there exists a sequence within the phrasal category list that matches one of the syntax structure rules. The output after applying the algorithm on list *Input* is a list of only one matched syntax structure rule, i.e. *Output = {(R4,1)}*.

Secondly the algorithm applies lexical extraction heuristics on a syntax structure rule-tagged sentence to extract content words for generating one or more predicates. For each identified syntax structure rule in a sentence the algorithm: (1) determines the position of both noun and verb phrases within the phrasal category sequence; (2) applies the heuristics to extract the content words (verbs and nouns) from each phrase category; (3) converts each verb and noun to its morphological root (e.g. *driving* to *drive*); and (4) generates the corresponding predicate p in the form *PredicateValue(Object1, Object2)* where *PredicateValue* is the verb and *Object1* and *Object2* the nouns. To illustrate this the algorithm identified rule *R4+* for our example sentence in Figure 5. According to one heuristic *{R4+}* corresponds to the following phrasal category sequence *[NP, VP, S, VP, NP]*. Therefore the algorithm determines the position of both noun and verb phrases within this sequence, i.e. noun phrases in *{NP,1}* and *{NP,5}* and verb phrases in *{VP,2}* and *{VP,4}*. Lexical extraction heuristics are applied to extract the content words from each phrase category, i.e. *{NP,1}* → *driver*, *{NP,5}* → *space*, *{VP,2}* → *need*, and *{VP,4}* → *find*. Returning to our example, the algorithm generates two predicates for the sentence *the driver needs to find a space in a car park close to his destination*, namely *need(driver,space)* and *find(driver,space)*.

3.3 The Predicate Expansion Component

Word mismatches are a problem in web service retrieval because analysts and service providers use different terms to describe use cases, requirements and web services [17]. In AnTiQue service queries are expanded using words with similar meaning. AnTiQue uses ontological information from VerbNet to extract semantically related verbs for verbs in each predicate.

AnTiQue's predicate expansion component uses members of (sub-)classes as potential expansion terms. All VerbNet (sub-)classes are organised so that there is syntactic and semantic coherence among members. For example the verb *book* as *in arrange for and reserve in advance* is one of 24 members of the *get* class. The list of members includes *buy*, *call*, *order*, *reserve*, etc. Thus VerbNet provides 23 verbs as potential expansions for the verb *book*. We constrain use of expansion to verb members that achieve a threshold on the degree of attributional similarity computed by applying a WordNet-based similarity measurement [16]. Given 2 sets of NL text, $T1$ and $T2$, the measurement determines how similar the meaning of $T1$ and $T2$ is scored between 0 and 1. For example, when considering the verb *book*, the algorithm computes the degree of attributional similarity between *book* and each co-member within the *get* class. In our example the accepted verbs such as *reserve*, *order* and *call* but not *reach* and *find* are used to generate additional predicates such as *call(x),* thus increasing the likelihood of retrieving relevant web service descriptions.

3.4 The Predicate Matcher

3.4.1 Coarse-grained Matching

Having generated a list of expanded predicates from the initial service query, all original and expanded predicate values are transformed into one or more XQueries that are fired at the web service registries. Prior to executing the XQueries we pre-process all web services in the registries using the Natural Language Processing and Predicate Parser components and store them locally. The XQueries include functions to match each original and expanded predicate value to equivalent representations of candidate web services.

SeCSE's service description facet in Figure 1 is structured using typed attributes such as *service goal*, *service actors* and *short service description* that service providers populate with relevant descriptions. AnTiQue uses these typed attributes to restrict term matching to equivalent typed attributes of service queries based on the structure of the original use case and requirement specification. Types in the query include *use case goals*, *use case actors* and *use case précis*, and the Predicate Matcher matches expanded predicate values from the use case précis to predicate values in the short service description.

3.4.2 Fine-grained Matching

The Predicate Matcher applies semantic and dependency-based similarity measures to assess the quality of the candidate web service set. It computes relational similarity between the service query and each web service retrieved during coarse-grain matching. To compute relational similarities that indicate analogical matches between service and query predicate arguments the Predicate Matcher uses the Dependency Thesaurus to select web services that are relationally similar to mapped predicates in the service query.

In our example the web service *Find Nearby Station*, which finds the location of nearby underground stations, is one candidate service retrieved during coarse-grained matching. The algorithm receives as inputs a pre-processed sentence list for a query (e.g. the précis) and service element (e.g. the short service description). It compares each predicate in the pre-processed query element sentence list $Pred(j)_{Query}$ with each predicate in the pre-processed service element sentence list $Pred(k)_{Service}$ to calculate the relevant match value, where

$Pred(j)_{Query} = PredVal_{Query}(Arg1_{Query}; Arg2_{Query})$

and

$Pred(k)_{Service} = PredVal_{Service}(Arg1_{Service}; Arg2_{Service})$.

The following conditions must be met in order to accept a match between the predicate pair:

1. $PredVal_{Service}$ exists in list of expanded predicate values of $PredVal_{Query}$;
2. $Arg1_{Query}$ and $Arg1_{Service}$ (or $Arg2_{Query}$ and $Arg2_{Service}$ respectively) are not the same;
3. $Arg1_{Service}$ (or $Arg2_{Service}$) exists in the Dependency Thesaurus result set when using $Arg1_{Query}$ (or $Arg2_{Query}$) as the query to the Thesaurus;
4. the resulting attributional similarity value from step 3 is below a specified threshold.

If all conditions are met, $Pred_{Service}$ is added to the list of matched predicates for the current web service. If not the algorithm rejects $Pred_{Service}$ and considers the next list item.

AnTiQue queries the Dependency Thesaurus to retrieve a list of dependent terms. Terms are grouped automatically according to their dependency-based similarity degree. Firstly the algorithm checks whether the service predicate argument exists in this list. If so, it uses the semantic similarity component to further refine and assess the quality of the service predicate with regards to relational similarity.

Using this 2-step process AnTiQue returns an ordered set of analogical services based on the match score with the service query. In our example consider $Pred(j)_{Query}$ = *find(driver,space)* extracted from the example sentence *the driver needs to find a space in a car park close to his destination,* and $Pred(k)_{Service}$ = *find(tourist,station)* extracted from the sentence *a tourist in London wants to find the nearest underground station* taken from the specification of the *Find Nearby Station* web service in Figure 1. In this example all 4 conditions are met:

1. Condition 1 is met since both predicate values are the same;
2. Condition 2 is met since *driver* and *tourist* as well as *space* and *station* are not the same;
3. Condition 3 is also met since *tourist* is similar based on dependencies to *driver, and station* is dependency similar to *space* (according to the Dependency Thesaurus);
4. Condition 4 is met since the attributional similarity value of *driver* and *tourist* is 0.25, for *space* and *station* 0.33 – both below the specified threshold.

Hence, the predicate *find(tourist,station)* is added to the list of matched predicates.

The next 2 sections report results from 2 evaluations of AnTiQue. We conducted these evaluations to seek answers to the 2 research questions about the precision, recall and usefulness of AnTiQue.

4. AnTiQue's Precision and Recall

The purpose of the first evaluation was to undertake a summative evaluation of the precision and recall of AnTiQue's algorithm and answer research question Q1 and explore whether AnTiQue could automatically retrieve analogical web services. The first evaluation was, in turn, divided into 2 studies – a human assessment of web services analogical to a specified use case, then an automatic assessment of the precision and recall of AnTiQue to retrieve analogical web services.

4.1 Similarity Classification of Web Service Descriptions

We used human judgment to determine which web services from a pre-selected set were analogical to car park booking, and which services were not analogical but similar to it in other ways. Firstly an expert in similarity research applied definitions for 4 different kinds of similarity – *literal similarity, analogy, mere appearance* and *anomaly* [1] – to generate 5 web service descriptions for each type of similarity to car park booking. One analogical web service *reserves hotel rooms*, a literally similar service *locates points of interest for a car driver* and a service that *plans walking routes for pedestrians* has appearance similarities unlikely to lead to effective reuse of the service. We then conducted a controlled study with 20 human judges – computer science researchers – who categorized the randomly ordered 20 web service descriptions based on similarities with car park booking. The categorizations, which judges made along continuous similarity scales, provided mean similarity values types for each web service for the judge group as whole, from which the results were generated.

Table 1 reports results. The judge group and similarity researcher agreed on the type of similarity for 16 of the 20 web services. Both identified 4 of the web services – for *cinema booking*, *hotel reservation*, *flight booking* and *train seat reservation* – as analogical to car park booking. However, unlike the researcher, the

judge group categorized the 5th analogical web service for *summer school booking* as an anomaly. The judge group and researcher also agreed on the categorizations of the 5 literally similar services and the 5 anomalous services that had no similarities with car park booking. In contrast the judge group and researcher only agreed that 2 of the 5 web services – *plan a walking route* and *compute journey distance time* – had mere appearance similarities with car park booking.

	Analog-ical	Literally similar	Mere appear-ance	Ano-ma-lies
Similarity researcher	5	5	5	5
Human judge group	4	5	2	5

Table 1. Totals of web services categorized by the similarity researcher and judge group by similarity type

These human judgments about the types of similarity between 16/20 web services and car park booking provided the baseline with which to assess AnTiQue. We investigated whether AnTiQue could retrieve the web services judged as analogical and not retrieve the web services judged as not analogical with car park use booking. To do this we measured the precision and recall of AnTiQue during service retrieval.

4.2 Evaluating the Precision and Recall of AnTiQue

We fired one query containing the use case précis in Figure 3 at the SeCSE service registry containing 215 existing web services in domains such as flight booking and the 20 web service descriptions judged by the judge group. AnTiQue retrieved 9 of the 235 services as analogical with car park booking. Totals of web services retrieved by similarity type are in Table 2.

Totals of web services	Literally similar	Ana-logi-cal	Mere appear-ance	Ano-ma-lies	Unclas-sified
In Registry	5	4	2	5	219
Retrieved	0	4	1	0	4

Table 2. Totals of web services retrieved by AnTiQue by similarity type

AnTiQue retrieved all 4 web services categorized as analogical by the judge group. It also retrieved the 5th service classified as analogical by the researcher but not the group, recorded as 1 of the 4 unclassified services in Table 2. Two other unclassified web services retrieved – *Find Nearby Station* and *Find Nearby Tourist Location* – had been part of the original 215 web services published previously. The similarity researcher agreed that both were also analogical with car park booking because of high relational and low attributional similarity between the generated predicates. Both services supported *users to find locations whilst moving*, similar to car park booking, but in syntactically different domains.

The 4th unclassified web service called *Fiat vehicle purchasing* and one mere appearance web service called *plan a walking route* retrieved were not analogical with car park booking.

Results were used to compute precision and recall scores for the query. Recall was defined as:

Total retrieved analogical services /
Total classified analogical services*100

AnTiQue retrieved all 4 analogical services, so the recall score was 100%. Precision was defined as:

Total retrieved analogical services /
Total discovered services*100

AnTiQue retrieved all 4 analogical services and 2 additional analogical services already published. Therefore the precision score was 66.6%.

Whilst the precision and recall scores for AnTiQue in the evaluation were good, the ordering of the retrieved web services on match scores was not. AnTiQue retrieved the web service *Fiat Vehicle Purchasing* with the highest match value, in spite of being categorized as similar to car park booking by mere appearance. The web service retrieved information about available vehicles in a region that the person then uses to produce a short-list.

We investigated the mappings between the relational predicates in the *car park booking* and *Fiat vehicle purchasing* descriptions computed by AnTiQue in Table 3. Similarities between the relational predicates (*driver,space)* and (*person,information)* computed using the verb *find* were consistent with the analogical match, as were similarities between the predicates (*driver,*)* and (*person,*)* computed using the verb *activate*. AnTiQue computed a third mapping between the relational predicates (*driver,*)* and (*vehicle,*)* also using the verb *find* shown in Table 3. However this mapping was inconsistent with the analogical match because *driver* is the operator of a vehicle and had a high degree of attributional similarity with *vehicle*. The mapping was therefore generated because condition 4 of fine-grained matching by the Predicate Matcher (section 3.4.2) computed a score (0.17) below the threshold for attributional similarity. This example highlights one potential limitation of computing the attributional similarity using WordNet-based similarity measures.

Target Predicates	Source PRedicates	Match Value
find(driver,space)	find(person,information)	2.36
activate(driver,*)	find(person,*)	1.5
find(driver,*)	find(vehicle,*)	1.82

Table 3. Matched predicates for *Fiat Vehicle Purchasing* service, where * indicates corresponding arguments that did not match

With overall confidence in the precision and recall of AnTiQue established, we investigated how analysts were able to discover requirements using retrieved analogical and literally similar web services to answer research question Q2 – can analysts use analogical services to discover requirements that they rank as more novel than requirements discovered from use case walkthroughs and literally similar web services?

5. Discovering Novel Requirements

Four analysts from Fiat in Torino specified requirements on the car park booking system in 2 phases: (i) in a use case walkthrough; (ii) in a walkthrough of web services retrieved by EDDiE and AnTiQue. Both walkthroughs took place in one workshop ran by the authors, one of whom facilitated the walkthroughs while the other operated UCaRE, EDDiE and AnTiQue on behalf of the analysts.

Each phase lasted 1 hour. In the first the facilitator walked the analysts through the use case précis then normal course to discover requirements for the car park booking system that the scribe documented in UCaRE. The walkthrough continued until the analysts were unable to discover more requirements. The result was a list of requirements $Req_{usecase}$. The scribe then generated a service query from the use case précis and searched the service registry described in section 4.2 using AnTiQue and EDDiE. AnTiQue retrieved 10 web services from which we retained the top 4 analogical ones S_{analog}, to use in the workshop. EDDiE retrieved 15 web services of which we retained the top 4 literally similar ones, S_{litSim}. We retained only the top web services to remain within the time available for the workshop.

In the second phase UCaRE presented the 8 retrieved web services in one list shown in Figure 6 that alternated analogical and literally similar services to avoid bias. The facilitator then walked the analysts through each web service to discover additional car park booking requirements that the scribe documented in UCaRE. The result was a list of requirements, $Req_{services}$. We defined requirements discovered using analogical services as Req_{analog} and requirements discovered using literally similar services as Req_{litSim}.

After the workshop the 4 analysts independently completed a questionnaire that rated each of the requirements in Req_{analog} and Req_{litSim} for appropriateness to car park booking on a simple 1-7 Likert scale.

5.1 Assessing Requirements Novelty

To assess the specified requirements for novelty in the car park booking domain we equated *novelty* to *dissimilarity* [11]. Requirements that score low similarities to requirements identified as prototypical of the domain were identified to be dissimilar and hence more novel. We identified 4 values of *Prot* with which to undertake a more sophisticated analysis of requirements novelty: (i) the requirements discovered from the first phase $Req_{usecase}$ generated by the analysts without any influence from the retrieved web services; (ii) the use case attributes that described the essential characteristics of car park booking; (iii) the use case normal course description of the important actions of the driver and service-centric system when booking a car park space; (iv) all of the text in (i), (ii) and (iii).

We defined

$$DSI = Domain\text{-}specific\ Information$$

and

$$Prot = DSI + Req_{usecase}$$

that is, the union of the domain-specific information and the requirements elicited prior to service discovery constitutes the target class of artefacts. We used a similarity measure to match both requirement result sets with *Prot* to compute the novelty score:

$$Sim_{litSim} = Similarity(Prot, Req_{litSim}) \in [0,1]$$
$$Sim_{analog} = Similarity(Prot, Req_{analog}) \in [0,1]$$

If the result is $Sim_{litSim} > Sim_{analog}$ then we show that analogical services trigger the discovery of more novel requirements. To compute similarity we compared both requirement sets Req_{analog} and Req_{litSim} with *Prot* using the WordNet-based semantic similarity measure [16] described in Section 3.

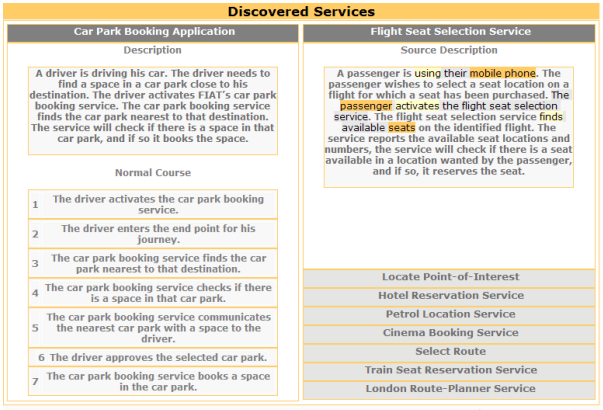

Figure 6. Retrieved service descriptions in UCaRE

5.2 Workshop Results

The analysts specified 61 requirements during the workshop. They specified 35 in the first phase and 26 in the second phase, 16 of which were generated from analogical web services Req_{analog}, and 10 from literally similar web services Req_{litSim}.

Figure 7 shows relative similarities between *Prot* and Req_{litSim} (Sim_{litSim}) and between *Prot* and Req_{analog} (Sim_{analog}). Each column depicts the average similarity

scores, converted into percentages, for requirements discovered from analogical and literally similar web services compared to the 4 different *Prot* values. Results revealed that the similarity between *Prot* and Req_{litSim} was, on average, higher than the similarity between *Prot* and Req_{analog}. Therefore we can conclude that is $Sim_{litSim} > Sim_{analog}$, and hence analogical web services triggered specification of some more novel requirements than did literally similar services.

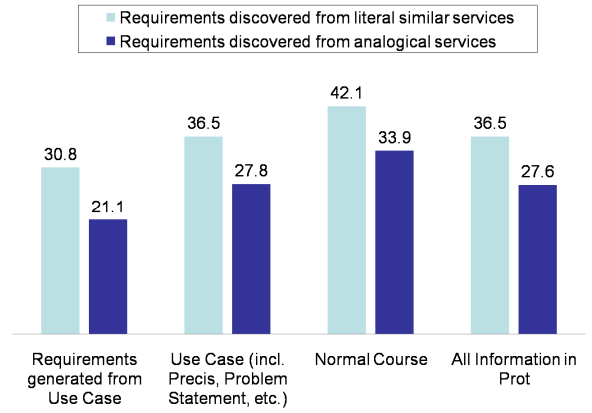

Figure 7. Similarity scores (in %) for requirements Req_{litSim} and Req_{analog} compared to 4 values of *Prot*.

Table 4 shows the average ratings per analyst of appropriateness of the 26 Req_{analog} and Req_{litSim} requirements specified in the second phase. Average ratings for the analysts show that Req_{analog} (4.5) were perceived as less appropriate to the target system than were Req_{litSim} (4.9), but this difference was insignificant.

Analyst	A1	A2	A3	A4
Requirements discovered from literally similar web services	5	5.3	5.6	3.8
Requirements discovered from analogical web services	4.56	5.25	4.25	4.13

Table 4. Average appropriateness ratings of requirements generated by each of the 4 analysts A1-A4 during the second phase, on a scale 1-7

6. Research Questions Revisited

We used results from the AnTiQue evaluations to answer the 2 research questions. The answer to the first question Q1 – can AnTiQue automatically retrieve web services from domains analogical to a specified problem – is yes, at least for the reported query and registry. From the natural language car park booking use case specification AnTiQue retrieved analogical web services also expressed in natural language with a recall score of 100% and precision score of 66.6% from a registry of 235 web service descriptions. However AnTiQue's fine-grain ordering of retrieved services on analogical match scores did incorrectly rank one non-analogical web service with the highest score.

The answer to the second question Q2 - can analysts reason with analogical services retrieved by AnTiQue to invent requirements ranked as more novel – was also yes, for the workshop. Analysts specified a greater number of requirements when reviewing web services for analogical domains than when reviewing web services that were literally similar to car park booking. Post-workshop analyses revealed that requirements specified when reviewing the analogical web services were more dissimilar to requirements and use cases specified prior to service retrieval with EDDiE and AnTiQue, and hence more novel according to the definition used. The absence of a significant difference in appropriateness rankings indicated that increased novelty did not come at the expense of the decreased usefulness of the requirements.

The results also provide evidence for the SeCSE iterative requirements process outlined in Section 2. The requirements generated from both analogical and literally similar web services indicated that analysts were able to discover new requirements by reviewing and working backwards from designs and implementations of services based on what might be possible.

Clearly there are threats to results validity. One threat to the conclusion validity of the evaluation results is the sample size – 1 service query from 1 use case specification fired at 1 registry and applied in 1 workshop. However the current small body of research into requirements techniques for service-centric systems (e.g. [14]) and the absence of any research into analogical services to encourage creative thinking led us to run a formative-predictive evaluation to generate a first set of results to explore AnTiQue's feasibility then provide a framework and focus for more subsequent rigorous evaluation.

A threat to the internal validity of the workshop results is the unintended bias from verbal guidance given by the facilitator and requirements writing undertaken by the scribe. Prior to the workshop the 4 analysts had experience with EDDiE but AnTiQue and its capabilities were unfamiliar, and research question Q2 was not made public. In contrast, whilst the facilitator used a protocol to guide interaction with the analysts both he and the scribe were aware of the research question, so implicit bias when guiding and documenting the analyst's work cannot be excluded.

Finally, one threat to the external validity of the results might have been the choice of domain. The results have external validity if we can generalize them outside of car park booking and analogies with it to other domains, so that available services might be retrieved analogically. We are unaware of research into problem domains for service-centric systems, but earlier requirements research of problem frames and domain models [18] indicates that widespread analogical reuse across domains is feasible.

7. Future Research on AnTiQue

The results provide a framework for future design and evaluation of AnTiQue. We plan to validate the results reported in this paper with larger-scale precision and recall experiments to learn whether AnTiQue can retrieve analogical web services across domains with different types of service query extracted from more than one use case specification. To do this we need to revise the Predicate Matcher's fine-grain matching algorithm to reduce the likelihood of incorrect attribute similarities leading to the retrieval of non-analogical web services. One option is to compute different attribute similarity measures with which to validate the WordNet-based similarity measure. We are also reviewing how the tools present analogical web services to stakeholders shown in Figure 6. Evidence from cognitive science [1] suggests that highlighted mappings between elements of text might not be as effective as showing graphical representations of mappings when transferring a analogical knowledge across 2 domains.

AnTiQue's success has implications for the SeCSE requirements process [2], in particular when to combine the use of AnTiQue and EDDiE to discover web services with different types of similarity to specify the requirements for a service-centric system.

Finally we are also interested to investigate whether analysts can work backwards to discover requirements from designs and implementations of software and design artifacts other than web services. Examples include commercial software documentation and reverse engineered specifications. We recently trialed UCaRE and EDDiE to support requirements reuse in a UK policing domain, and plan to report results shortly.

8. Acknowledgements

The research reported in this paper is supported by the EU-funded 511680 SeCSE Integrated Project.

9. References

[1] Gentner D., 1983, 'Structure-mapping: A theoretical framework for analogy', Cognitive Science, 7, 155-170.

[2] Jones S.V., Maiden N.A.M., Zachos K. & Zhu X., 2005, 'How Service-Centric Systems Change the Requirements Process', Proceedings REFSQ'2005 Workshop, CAiSE'2005, 13-14 June 2005, Porto, Portugal.

[3] Kipper K., Dang H.T., & Palmer M., 2000, 'Class-based construction of a verb lexicon', AAAI-2000 Seventeenth National Conference on Artificial Intelligence, Austin, TX, USA.

[4] Leavitt, N., 2004, 'Are Web Services finally ready to Deliver?', IEEE Computer, 37, 1418.

[5] Levin B., 1993, 'English Verb Classes and Alternations: A Preliminary Investigation', University Chicago Press.

[6] Lin D., 1998, 'Automatic retrieval and clustering of similar words', In COLINGACL, 768-774.

[7] Maiden N.A.M. & Sutcliffe A., 1992, 'Exploiting reusable specifications through analogy', Communications of the ACM, 35, 55-64.

[8] Maiden N.A.M., Gizikis A. & Robertson S., 2004, 'Provoking creativity: Imagine what your requirements could be like', IEEE Software, 21, 68–75.

[9] Maiden N.A.M., Ncube C. & Robertson S., 2007, 'Can Requirements Be Creative? Experiences with an Enhanced Air Space Management System', Proceedings 28th International Conference on Software Engineering, ACM Press, 632-641.

[10] Nuseibeh B. & Easterbrook S., 2000, 'Requirements Engineering: A Roadmap', Proceedings IEEE International Conference on Software Engineering (ICSE-2000), 4-11 June 2000, Limerick, Ireland, ACM Press.

[11] Ritchie G., 2001, 'Assessing Creativity', Proceedings AISB-01 Symposium AI and Creativity in Arts and Science.

[12] Robertson J., 2005, 'Requirements analysts must also be inventors', IEEE Software, 22, 48–50.

[13] Robertson S. & Robertson J., 1999, 'Mastering the Requirements Process', Addison-Wesley-Longman, Redwood City.

[14] Rolland C., Kaabi R.S., Kraiem N., 2007, 'On ISOA: Intentional Services-Oriented Architecture', Proceedings CAiSE'2007, 13-15 June 2007, Trondheim, Norway, 158-172.

[15] Sawyer P., Hutchinson J., Walkerdine J. & Sommerville I., 2005, 'Faceted Service Specification', Proceedings SOCCER (Service-Oriented Computing: Consequences for Engineering Requirements) Workshop, at RE'05 Conference, Paris, August 2005.

[16] Simpson T. & Dao T., 2005, 'Wordnet-based semantic similarity measurement', codeproject.com/cs/library/semanticsimilaritywordnet.asp.

[17] Singhal A. & Pereira F., 1999, 'Document expansion for speech retrieval', In Proceedings of ACM SIGIR, 3441, Berkeley, CA, USA.

[18] Sutcliffe A.G. & Maiden N.A.M., 1998, 'The Domain Theory for Requirements Engineering, IEEE Transactions on Software Engineering, 24(3), 174-196

[19] Tetlow P., Pan J., Oberle D., Wallace E., Uschold M. & Kendall E., 2005, 'Ontology Driven Architectures and Potential Uses of the Semantic Web in Software Engineering', W3C (2005).

[20] WordNet 2005, Version 2.1, http://www.cogsci.princeton.edu/wn.

[21] Zachos K., Maiden N., Zhu X., & Jones S., 2006, 'Does Service Discovery Enhance Requirements Specification? A Preliminary Empirical Investigation', Proceedings SOCCER (Service-Oriented Computing: Consequences for Engineering Requirements) Workshop, a t RE'06 Conference, Minneapolis, September 2006.

[22] Zachos K., Maiden N.A.M., Zhu X. and Jones S.V., 2005, 'Discovering Web Services to Specify More Complete System Requirements', Proceedings CAiSE'2007, 13-15 June 2007, Trondheim, Norway.