The benefits of domain models for checking the completeness of requirements are known, but have
never been evaluated systematically. We focus on requirements written as "shall"-style statements
and domain models captured using UML class diagrams. We observe that domain models exhibit near-
linear sensitivity to both unspec-ified (i.e., missing) and under-specified requirements.
A fundamental property of external completeness is that it cannot be ascertained in absolute terms.
Despite this limitation, one can implement measures for improving (but not guaranteeing) the
external completion of requirements. One such measure is domain modeling (Zowghi and Gervasi 2003a;
Ferrari et al. 2014). A domain model is an explicit representation of the salient concepts in an
appli- cation domain.
Checking the completeness of NL requirements is partic- ularly challenging, noting that NL
requirements specifications may constitute hundreds orthousands of statements. This makes it
important to develop a more detailed understanding of mechanisms through which one can identify
incompleteness issues in NL requirements. One interesting mechanism to study for this purpose is
domain modeling.
The requirements in Fig. 1a have been slightly altered from their original form to facilitate
discussion and preserve confidentiality. The requirements document from which this subset was taken
is the subject of one Fig.1 aExamples of shall requirements; bExample domain model represented as a
UML class diagram.
An analyst can use a reasonably complete domain model for detect- ing incompleteness in
requirements. For example, the model of Fig. 1does not convey the constraint stated in REQ1 that the
simulations shall be configured "via a web service" We note again that we are arguing under the
assumption that the domain modelis complete.
In Fig. 1, domain concepts may be referred to several times in the requirements. The higher the
level of repetition, the less sensitive the domain modelbecomes to omissions. Most importantly, we
observe that domain models exhibit near-linear sensitivity to both unspecified requirements and
requirementsthat are missing.
 domain models are sufficiently sensitive toomissions in NL requirements. The level of sensitivity
that domain models show to omissions is more than four 2514
EmpiricalSoftwareEngineering(2019)24:2509–2539 times higher when whole requirements are removed.

Despite being promising, our results do not automatically lead to the conclusion that domain model
are useful instruments for completeness checking of requirements by ana-lysts.
Backward complete- ness is measured against the body of knowledge established prior to the development of a requirements specification. Forward completeness is, in contrast,measured against
the (accumulated) body ofknowledge after a requirement specification has been produced. Our mode of
using domain models thus coincides with forward checking of NL requirements.
Lindland et al. (1994) introduce the notion of feasible completeness, where a con- ceptual model
contains only a subset of the statements in a domain. A conceptual modelis feasibly complete with
respect to a domain, if any further enhancement of the model isdeemed less beneficial than accepting
the model as-is. In our case studies, we aim to achieve this completeness for the domain models
built.
Using NLP, Gigante et al. (2015) extract from a set of requirements and an external source – in
their case, higher-level requirements – sys-tematic information in the form of (subject ,predicate,
object)triplets. They then compare the triplets from the two sources to determine how complete the
given set of requirementis. No statistically- significant results are obtained for completeness.
We examine how a complementary approach, namely domain modeling, contributes toimproving
requirements completeness. The main components of our study are: (1) constructing feasibly complete
domain mod- els, (2) establishing traceability between the domain models and the requirements, and
(3) simulating potential omissions in the requirements.
The study looked at the use- fulness of domain models for completeness checking of requirements. The
data collection was aimed at constructing (feasibly complete) domain models, defining potential but
realistic omissions from requirements, tracing requirements to domain model elements, and gathering
data to argue about the repre-sentativeness of the results.
In each case study, we randomly selected 35 requirements. We extracted all the atomic noun phrases
(NPs) from the sentences in R. Following object-oriented domain modeling guidelines (Larman 2004),
we considered each NP as a candidate domain con- cept. Without revealing Rto the domain experts, we
had them review the NPs extracted from these requirements.
Two researchers (the first two authors) acted as facilitators during modeling. In all three case
stud-ies, the experts did a portion of the modeling work offline but collaboratively. The

modeling
activities concentrated on specifying thecore domain model elements.

To simulate the omission of requirement segments in a realistic manner, both semantic interdependencies and the significance of the content of segments need to be considered. The main criterion to decide whether a segment was omittable was the plausibility of thesegment being overlooked in realistic conditions.

Omittable segments fall into one of the following types: Object, Condition, or Constraint. Objects are, in principle, omittable only when several of them are present. Constraints restrict thedesign or implementation of the systems engineering process.

3.3 TracingRequirementstotheDomainModel In the last step of data collection, we had the experts in each case study trace the selectedrequirements, denoted Rearlier, to the respective domain model.

Recall from Section 3.3.1 that each domain model was built around the set of concepts derived from R.

Fig. 4 shows the union of all trace links originating from a given omittable segment. In our example of Fig. 4, REQ2 has a condition, Condition 2 1. The term 'simulation' in Condition21is traceable to the domain model. This induces a trace link from Condition2 1toSimulation.

We present our simulation algorithm in Fig 5. The inputs to the algorithm are: (1) the outcomes of our data collection procedure, described in Section 3.3, (2) the type of omissions to simulate (T), and (3) the number of simulation runs (n) Following a randomized process, the algorithm seeds into requirements progressively larger sets of omission of a given type.

Table 1 provides key statistics about the outcomes of our data collection as per the proce- dures discussed in Section 3.3.1. Table 2provides statistics. about the complementary data that we gathered according to the procedure described in Section. 3.4 and with the goal of examining the representative- ness of our case studies. The scatter plot resulting from running the algorithmof Fig. 5is the basis for answering RQ1 in Section 4.

Fig. 6shows the output (scatter plots) for n=100 runs of our simulation algo- rithm (Fig. 5) Each plot shows, for each case study, the relationship between the number of omissions of a certain type and the percentage of domain model elements that are no longer supported as the result of the omissions. The amount of time the researchers spent with the experts for data collection is 8 hours

for each Case A and Case B, and 6 hours for Case C.

In Case C, the domain conceptsappear considerably more frequently in the requirements than in Case A

and Case B. Figure 6b, c and d show the sensitivity of the domain models in our case studies to

omis- sions of conditions, constraints, and objects, respectively.

The level of sensitivity is, on average, 4.4 times higher when require- ments are removed in their

entirety than when conditions, constraints and objects areremoved. The impact of omitting 10

requirements from Case A is the loss of support for 23.4% of the domain model elements.

Using keyphrases as a surrogate for domain model elements is most practical for pre- dicting

sensitivity to requirement omissions. NPs and VPs are the main meaning-carrying units of requirements statements (Arora et al.2017). Determining whether a keyphrase is still supported after

a set of omissions is trivial.

The Pearson's correlation coefficient (Warner 2012) for Case A. is 0.97440 ($p < 0.0001$), for Case B.

is. 0.98005 ($p = 0.00001$), and for Case C is.0.96838 ($p > 0.0005$) Figure 9 shows the keyphrase

sensitivity curves for both the requirements subsets and the full documents, organized by case

study.

The main threat to the internal validity of our work is subjectivity in mod- eling. To minimize

subjectivity about content, we engaged closely with at least two domain experts in each case study.

We believe that other experts, under identical circumstances, would have made sim-ilar choices about

content. In measuring sensitivity, we are concerned only with whether domain models contain the

information that is necessary for revealing omissions.

The study is not meant at measuring theoverhead associated with investigating such false alarms. The

above overhead appears to be a reasonable price to pay for the genuine requirementsincompleteness

issues that one can identify through a domain model.

This article was motivated by the anecdotal observation that domain models are additionallyhelpful

for detecting incompleteness in requirements. To empirically examine this observa-tion, we conducted

three industrial case studies with subject-matter experts. In each casestudy, we measured the

incomple completeness-revealing-power of a domain model by seeding realistic omissions into the

requirements.

Arora C, Sabetzadeh M, Briand L, Zimmer F (2015) Automated checking of conformance to requirements

templates using natural language processing. Ambler S (2004) The object primer: agile model-driven
development with UML 2.0, 3rd edn. Cambridge University Press, Cambridge Apache OpenNLP Chunker.
Epa ˜na S, Condori-Fernandez N, Gonzalez A, Pastor ´O (2009) Evaluating the completeness and granu-
larity of functional requirements specifications: a controlled experiment. Femmer H, Mager P (2016)
Challenging incompleteness of performance requirements by sentence patterns.
Error during summarization.
Error during summarization.
Error during summarization.
Error during summarization.