This paper investigates the potential usefulness of domain models for detecting incompleteness in natural-language requirements. The study focuses on requirements written as "shall"-style statements and domain models captured using UML class diagrams. Through a randomized simulation process in an industrial setting, experts constructed domain models in three industry domains to analyze the sensitivity of domain models to omissions in requirements. The research found that domain models exhibit near-linear sensitivity to unspecified and under-specified requirements, with a higher sensitivity for unspecified requirements. The results suggest that domain models can provide cues for checking the completeness of requirements, though further studies are needed to determine if analysts can effectively use these cues for detecting incompleteness.

The text discusses the potential use of domain models for detecting incompleteness in natural-language requirements. It highlights the challenge of checking the completeness of NL requirements due to their extensive nature, and suggests that domain modeling could provide a mechanism to identify incompleteness issues. The study focuses on shall requirements and UML class diagrams as representations for NL requirements and domain models, respectively. The research question addressed is whether UML class diagrams can be useful for checking the completeness of functional requirements expressed as shall statements. An example is provided to illustrate how a domain model can assist in checking the completeness of requirements by showing a subset of requirements for a simulator tool in the aerospace domain alongside a domain model fragment.

The text discusses how domain models and requirements may contain different information, with some details only present in one or the other. The authors argue that just because additional information can be found in the domain model compared to the requirements, it does not necessarily mean the requirements are incomplete. The text also explains how analysts can use domain models to detect incompleteness in requirements, such as by cross-checking for missing concepts or associations. The level of repetition of domain concepts in the requirements can impact the sensitivity of the domain model to omissions.

The text discusses the importance of domain models in checking the completeness of natural language

(NL) requirements. Two main factors affecting the usefulness of a domain model for this purpose are identified: the level of content overlap between requirements and the domain model, and how frequently the domain model elements are referred to in the requirements. The sensitivity of domain models to omissions in requirements is empirically investigated through three case studies in different application domains. The study shows that domain models exhibit near-linear sensitivity to unspecified and under-specified requirements. However, further research is needed to determine if analysts can successfully detect omissions using domain models and if the extra information provided by domain models is essential for completing requirements. The text also outlines related research on requirements completeness and discusses the structure of the paper.

The text discusses the importance of checking the completeness of requirements in software engineering. It distinguishes between internal and external completeness, with external completeness further divided into backward and forward completeness. The concept of completeness is also applied to conceptual models, with a focus on feasible completeness. Various strategies for checking the completeness of requirements are outlined, including using structured templates, engaging stakeholders, employing formal synthesis and verification, cross-validating requirements against other artifacts, and conducting reviews and inspections. The text emphasizes the significance of ensuring that requirements are complete to serve as the contractual basis for system development.

The text reviews various approaches to checking the completeness of natural language requirements, focusing on external completeness. Different methods include assessing whether high-level requirements are decomposed, using natural language processing (NLP) to compare terminology, employing NLP and visualization to extract and contrast concepts, mapping requirements concepts to a domain ontology, extracting systematic information from requirements, and comparing lightweight models against best practices. The text highlights the importance of using a domain model for completeness checking and emphasizes the need for empirical evaluation. Controlled experiments comparing different requirements specification approaches have been conducted, with varying results on completeness. The text introduces a study that focuses on shall requirements and UML class diagrams to improve requirements completeness through domain modeling in an industrial

setting.
The text discusses three case studies: Case A, involving a simulator module for aerospace applications; Case B, a sensor platform for cyber-physical systems; and Case C, a content management
system for safety assurance. The study focuses on constructing domain models, establishing traceability between domain models and requirements, and simulating potential omissions in requirements. The research questions address the sensitivity of domain models to requirement
omissions and how this sensitivity relates to the intrinsic properties of requirements documents.
Case selection criteria included finalized requirements documents, access to domain experts, and
coverage of different domains. Data collection involved constructing domain models, defining
omissions from requirements, tracing requirements, and ensuring representativeness of results.
Expert involvement in data collection was significant, with experts having domain experience and
familiarity with UML.
The academic text describes a methodology for constructing domain models based on selected requirements and expert input. The process involves extracting noun phrases from requirements,
having domain experts review and refine them, and then creating a domain model focusing on the
selected concepts. The experts are allowed to introduce additional concepts as needed. The domain
modeling process follows object-oriented analysis best practices and involves specifying core
elements like classes, attributes, and relations. The experts were guided by researchers and used
existing sketches as a starting point for modeling. The text also discusses simulating requirements
omissions by omitting entire requirements or segments of individual requirements. The domain
modeling was completed before further data collection to avoid bias.
The text discusses the importance of semantic interdependencies in making sentences coherent and
meaningful. Researchers identified omittable segments in requirements by considering semantic types
and interdependencies. Interrater agreement was used to ensure reliability. Omittable segments were
reviewed and approved by experts. The text also explains the types of semantic slots defined by the
IEEE 29148:2011 standard. The findings of interdependencies between omittable parts are presented as
part of the results. Three case studies were conducted to analyze omittable segments, interdependencies, and agreement among researchers and experts.
The text discusses the identification of omittable segments in software requirements, specifically

focusing on conditions and constraints. Three possibilities for constraints are outlined: (a) no interdependencies between segments, allowing any subset to be omitted independently, (b) at least one segment must be retained, and (c) if one segment is removed, another related segment must also be removed. The text also highlights the tracing of requirements to domain models, emphasizing the importance of experts in establishing links between requirements and domain model elements.

Additionally, the text mentions gathering complementary data to analyze the representativeness of the results obtained from a fraction of the total requirements documents.

The researchers collaborated to identify interdependencies between omittable segments in requirements documents by marking up keyphrases using NLP technology. The analysis procedure involved a Monte-Carlo simulation algorithm to simulate different types of omissions and assess the impact on domain model elements. Results were presented in a scatter plot showing the percentage of unsupported domain model elements as omissions increased. The study demonstrated the sensitivity of the domain model to omissions and the implications for tacit domain model elements in requirements.

The text discusses the results and analysis of case studies conducted to examine the representativeness of requirements samples. Key statistics, including the number of omittable elements, inter-dependencies, interrater agreement, domain model elements, and trace links, are presented in Tables 1 and 2. The simulation algorithm used in the study is shown in Figure 5, with scatter plots illustrating the relationship between omissions of certain types and the percentage of unsupported domain model elements. The findings suggest that Case C exhibits lower sensitivity to omitted requirements compared to Cases A and B, indicating that more omissions are needed in Case C for domain model elements to lose support. Various influencing factors contributing to this difference are mentioned.

The text discusses the sensitivity of domain models to omissions in requirements, focusing on three different cases (Case A, Case B, and Case C). It highlights that the frequency of appearance of domain concepts in the requirements impacts the sensitivity of the domain model to omissions. Case C stands out as having domain concepts appearing more frequently in the requirements compared to Case A and Case B, resulting in lower sensitivity to requirement omissions. The study shows

that domain
models are more sensitive to unspecified requirements than under-specified requirements. Additionally, the text compares the impact of omitting requirements, conditions, constraints, and
objects on the support of domain model elements, with whole requirement omissions having a significantly larger impact. The study concludes that domain models exhibit near-linear sensitivity
to requirement omissions, with Case A being the most sensitive and Case C being the least sensitive
to requirement omissions.
The text discusses how to determine the sensitivity of a domain model to different omissions in
requirements. It highlights the importance of predicting sensitivity based on characteristics of a
requirements document, such as the frequency of terms. The text introduces using unsupported
keyphrases as a surrogate for predicting sensitivity and proposes a simulation algorithm to assess
this. Keyphrases are considered a practical predictor for sensitivity to requirement omissions,
while building a predictor for constraint omissions is deemed less practical. The text also
discusses the representativeness of the analysis and concludes that keyphrases can be a good proxy
for measuring sensitivity to omissions in domain models.
The study compares the sensitivity of keyphrases to different types of omissions in full requirements documents versus subsets. The results suggest that the characteristics of the full
documents are similar to those of the subsets used in the case studies. The study concludes that the
case studies are reasonably representative of the full requirements documents, providing confidence
that the sensitivity levels observed would generalize to the full documents. The text also discusses
threats to validity, including subjectivity in modeling and the importance of domain experts in
minimizing subjectivity. It further notes that the study focuses on identifying omissions in
requirements rather than measuring the overhead associated with false alarms. The study does not
assess the cost-benefit tradeoffs of using domain models for completeness checking but highlights
the value of identifying genuine requirements incompleteness issues.
The text discusses the potential of domain models for revealing omissions in requirements and the
challenges related to construct validity and external validity. The study conducted three case
studies to examine the ability of domain models to detect incompleteness in requirements. It

emphasizes the importance of building domain models with diligence and suggests further investigation into the effectiveness of domain models constructed under time and cost constraints.

Future work includes user studies to assess how analysts can utilize domain models for finding incompleteness issues and exploring the use of models for non-functional requirements. The study was funded by the Luxembourg National Research Fund and the European Research Council. 

The text contains references to various academic papers and resources related to software requirements engineering and natural language processing. It covers topics such as automated checking of requirements conformance, extraction and clustering of requirements glossary terms, evaluation of software requirements documents, and completeness and granularity of functional requirements specifications. Additionally, it includes information on methodologies and tools for modeling functional structures, verifying and validating requirements, and improving the completeness of natural language requirements.

The text discusses various academic papers related to software requirements analysis and engineering. Topics covered include real-time process-control systems, natural language processing, ontology-based requirements analysis, improving requirements quality, viewpoint merging, and the influence of requirements specification notation on change impact analysis. Additionally, the text addresses the completeness of functional requirements specifications, market research for requirements analysis, and the quality of entity relationship models. Other topics include expressing relationships between multiple views in requirements specification, software requirements inspections, fault detection in user requirements documents, and guidelines for enhancing the quality of information models.

The academic text discusses the research interests and affiliations of three researchers: Chetan Arora, Mehrdad Sabetzadeh, and Lionel C. Briand. Chetan Arora is an Industrial Research Fellow with expertise in requirements engineering, empirical software engineering, and applied machine learning.

Mehrdad Sabetzadeh is a Senior Research Scientist with a focus on software engineering, particularly requirements engineering, model-based development, and regulatory compliance. Lionel C. Briand is a professor in software verification and validation with research interests in software testing and verification, model-driven software development, search-based software engineering, and empirical

software engineering.