Chapter 2
State-of-the-Art

Several languages and frameworks have been proposed in the domain of goal oriented requirements engineering that try to capture and model the requirement specifications of the system being developed. Some well documented articles have been published that compare and contrast these approaches and stress on the analytical capabilities of each approach [1–4]. For the purpose of this book, we provide a brief summary of the current state-of-the-art in the requirements engineering domain in Sect. 2.1.

The rest of this chapter is organized based on the structure of our book. Section 2.2 presents a review of some of the existing works on ontology integration and highlights how semantic integration of different levels of a requirement refinement hierarchy has not yet been addressed by the research community. This section also presents a study of different data mining techniques that have been applied in the domain of requirements engineering. Based on these reviews we present our work on i* refinement hierarchies and mining goal decomposition patterns in Chapter 3. Chapter 4 of this book presents an efficient solution for model checking goal models like i*. An extensive background study related to model transformation techniques has been documented in Sect. 2.3. Based on this study, we propose a model transformation scheme that allows analysts to perform model checking on i* models. We also present a survey of goal model annotation nomenclatures that have been proposed as part of the current state-of-the-art in Sect. 2.4. This survey forms the basis for the AFSR framework proposed in Chapter 5 which proposes a new goal model nomenclature for capturing the semantics of goal models. Each of these sections conclude with a gap analysis that helps in identifying the research question that we address in the subsequent chapters of this book.

N. Deb and N. Chaki, *Business Standard Compliance and Requirements Validation Using Goal Models*, Services and Business Process Reengineering,
https://doi.org/10.1007/978-981-15-2501-8_2

2.1 Formal Requirements Engineering Techniques

One of the earliest requirements modelling language, Structured Analysis and Design Technique (SADT) [5], was proposed by Ross and Schoman in 1977. The language was founded on the principle of data/operation duality where data were defined by their source and destination operations while operations were defined on the basis of their input and output data. The main drawbacks of this early modelling language was the lack of precision and the absence of well-defined semantics. The syntax was also semi-formal and often interleaved with natural language assertions.

The first requirements modelling language that incorporated formal semantics was RML [6]. It introduced the semantic concept of entities, operations and constraints. Operations and constraints were expressed in formal assertion languages that supported temporal ordering of events. RML also introduced the concepts of generalization, aggregation, and classification. These formal semantics were mapped to first order predicate calculus.

Albert II [7] is a requirements language that was proposed with richer ontologies and modelled the requirements of agent-oriented systems. Entities were replaced by agents and modelled using graphical notations. Constraints on agent behavior were still modelled using textual notations that supported temporal logistics. Verification of constraints using formal analysis techniques, like animation, were possible. The drawback of this language was that it did not support enough abstraction for being used in the early phases of requirements engineering.

The state-of-the-art in requirements modelling focuses more on goal-oriented ontologies that capture the "*why*" requirements of an enterprise. The NFR framework [8, 9] specializes in the modelling and analysis of non-functional requirements or softgoals. The NFR framework deals with capturing non-functional requirements (NFRs) for the domain of interest, decomposing NFRs, identifying possible NFR operationalizations, NFR ambiguities, trade-offs, priorities, and NFR interdependencies [1].

Lamsweerde proposed the Knowledge Acquisition in autOmated Specification (KAOS) [4] framework that supports semi-formal modelling of goals, qualitative analysis of alternatives, and formal analysis for correct reasoning. The KAOS framework combines semantic nets [6], for modelling of concepts, and linear-time temporal logic for state-based specification of operations. Operations are declared by signatures over objects and have pre-, post-, and trigger conditions [1]. Goals can be decomposed using AND/OR refinement abstraction hierarchies. The KAOS framework does not provide any support for non-functional requirements or softgoals. However, the qualitative analysis techniques of the NFR framework can be integrated into KAOS. It has a solid formal framework that uses well-established formal techniques for goal refinement and operationalization [10, 11].

The Goal-Based Requirements Analysis Method (GBRAM) [12, 13] emphasizes on the identification and elicitation of goals from various documents as provided by the stakeholders of an enterprise. GBRAM distinguishes between achievement and maintenance goals. GBRAM tries to establish an ordering of goals in order to

establish inter-actor dependencies. This process is much more complex in comparison to the i* framework which can capture such dependencies quite efficiently. GBRAM does not provide a graphical interface; rather, it uses a textual notations in goal schemas for representing goals, goal refinements, goal precedence, agents, etc.

i* [2, 14] is an agent-oriented modelling framework that can be used for requirements engineering, business process re-engineering, organizational impact analysis, and software process modelling [1]. This i* framework can model activities prior to the freezing of requirement specifications. This allows the use of i* for both the early and late phases of requirements engineering. In the early phases, the i* model can capture the stakeholders of the enterprise, their objectives and how they depend on each other for achieving their objectives. The late phases of requirements engineering can use the i* modelling framework to incorporate changes and new processes that are aligned with the functional and non-functional requirements of the user.

2.2 Requirement Refinement Hierarchies

Different stakeholders within an enterprise may use partial or even completely non-intersecting vocabulary sets. This would lead to developing multiple i* models having independent ontologies. Collectively, these i* models define a requirements refinement hierarchy where different tiers of the hierarchy capture different levels of detail. In order to integrate such a distributed ontology, we require an appropriate mapping or correlate definitions between different ontologies at multiple levels. This section explores the ontology integration mechanisms that exist in the current state-of-the-art. We also try to document the research initiatives that have been taken in the last decade for applying data mining techniques in requirements engineering.

Ontology Integration Mechanisms

Graph matching of any structure is a well researched problem and several good works (like [15]) have been published in this domain. However, the real challenge being addressed here is conceptual matching of ontologies. Wang et al. [16] propose a tree similarity algorithm for ontology integration of multiple information sources available in the Semantic Web. Dynamic programming is used to effectively match concept trees while satisfying the maximum mapping theorem and keeping tree isomorphisms intact. PRIOR+ is an adaptive ontology mapping approach proposed in [17]. It consists of an information retrieval system that extracts similarities between ontologies, an adaptive similarity filter that aggregates these similarities, and, finally, a neural network based ontology constraint satisfier.

In [18], the authors bridge the gap between Description Logic based ontologies and Object Oriented systems by mapping OWL ontologies to Java interfaces and classes. STROMA [19] is a semantic ontology mapping scheme that goes beyond equality correlations and maps part-whole as well as IS_A relationships. Khattak et al. have highlighted how ontologies evolve over time [20]. A mapping reconciliation mechanism for evolving ontologies is proposed that reduces the

reconciliation time by tracking the change histories of such ontologies. [21] proposes relation mappings between ontologies by finding the least upper bound and greatest lower bounds of complex relations and then deriving the best upper and lower approximations of the relation. Kumar and Harding [22] use Description Logic based bridging rules for mapping complex concepts and roles between manufacturing and marketing enterprises. The Semantic Bridge Ontology [23] detects structural and semantic conflicts between Learning Resource Systems and resolves them by defining ontology mapping rules.

Data Mining in RE

Zawawy et al. have proposed a root-cause analysis framework [24] that mines natively generated log data to establish the relationship between a requirement and the pre- and post-conditions associated with that requirement. In [25], the authors have proposed techniques for mining dependencies from message logs and task-dependency correlations from process logs. There have been very interesting industrial and commercial applications of mining requirements from event logs. Formal verification of control systems have been performed by mining temporal requirements from simulation traces [26]. Qi et al. have provided big data commerce solutions by mining customer requirements from online reviews and suggest product improvement strategies [27]. REQAnalytics [28], proposed by Garcia and Paiva, mines the usage statistics of a website and provides a roadmap for the evolution of the website's requirements specification. [29] is another data mining technique that tries to address the inconsistencies that affect the contextual requirements of a system at runtime.

Sequential pattern mining has been frequently used for extracting statistically relevant patterns or sequences of values in data sets. StrProM [30], for instance, uses the Heuristics Miner algorithm to generate prefix-trees from the data stream and continuously prunes these trees to extract sequences of events. Sohrabi and Ghods use bitwise compression techniques to represent the data sequence as a 3-dimensional array and extract frequently occurring patterns from this compressed array [31]. Hassani et al. have proposed the PIVOTMiner [32] which considers activities as interval-based events rather than the conventional single-point events. Some researchers have also tried to improve the legacy sequential mining algorithm PrefixSpan (like [33–35]). Sequential pattern mining has also been used in interesting applications that range from detecting user behavior from online surveys [36] to mining electronic medical records and inferring the efficacy of medicines [37]. A detailed survey of sequential pattern mining algorithms is available in [38].

Previously workflow logs used to be mined for extracting the control flow within an organization and, hence, extensively used for developing process models. However, the mining process had no focus on extracting the hierarchical structure within an organization. Ni et al. have introduced the concept of executor similarity metrics and grid clustering for mining the organization structure of an enterprise from workflow logs [39]. Schönig and his group have proposed a framework to extract the organisational structure of business processes by mining human resource allocation information from event logs [40]. Also in prior work, non-functional requirements have been extracted from text [41].

Research Gap

A vast literature exists for ontology mappings within the domain of Semantic Web. However, there has been limited research on mapping model constructs between conceptual models derived from an enterprise hierarchy that use different ontologies. This research gap is identified and some ideas in the direction of bridging ontologies within hierarchic i* models developed by any enterprise is presented in Chapter 3.

2.3 Model Checking with i*

The importance of converting an i* model to a finite state model lies in the effort to perform model checking on i* models using industry standard model checkers like NuSMV and SPIN. Model checking helps requirement analysts to verify whether goal models comply with system regulations. Standard model checkers accept extended finite state models as input and verify temporal properties specified using Linear Temporal Logic (LTL) or Computational Tree Logic (CTL). i* models can be converted to other sequential models like activity diagrams or BPMNs for some specific business requirement, if the need arises. Transforming finite state models to other sequential models should be easier than transforming a sequence agnostic model, like i*, to activity diagrams or BPMNs. Model transformation represents the daunting challenge of converting higher-level abstraction models to platform-specific implementation models that may be used for automated code generation.

Model Transformation

Sendall and Kozaczynski had already identified model transformation as one of the major driving forces behind model-driven software development [42]. Most strategies work with lower levels of abstraction and encounter several limitations. In [43], the authors propose a Domain Specific Language over Coloured Petri-Nets—called Transformation Nets—that provides a high level of model transformation abstraction. An integrated view of places, transitions, and tokens, provide a clear insight into the previously hidden operational semantics.

Model transformation plays a vital role in bridging the gap between non-successive phases of the software development life cycle. [44] presents one such attempt to bridge the gap between system designers and system analysts. A model generated by the designer is transformed to a model suitable for conducting analysis. The outcome of the analysis is mapped back into the design domain. The authors work with *UML2Alloy*—a tool that takes a UML Class diagram augmented with OCL constraints and converts it into the Alloy formal representation. Design inconsistency analysis is done on the Alloy representation. Alloy creates counter examples for any such inconsistency and converts it back into a UML Object diagram. This paper tries to do model transformation for bridging the gap between the requirements phase and the design phase of the development life cycle.

Creating a wide array of formal models for enhancing the system engineering process, proves to have time and cost overheads. Kerzhner and Paredis use model transformations to achieve this objective, overcoming the overheads, in [45]. Formal models are used to specify the structures of varying design alternatives and design requirements, along with experiments that conform the two. These models are represented using the Object Management Group's Systems Modelling Language (OMG SysMLTM). Model transformation is then used to transform design structures into analysis models by combining the knowledge of reusable model libraries. Analysis models are transformed into executable simulations which help in identifying possible system alternatives. Model transformation plays a vital role in this work.

Mussbacher et al., have performed a detailed comparison of six different modelling approaches in [46]. The modelling approaches that were assessed include Aspect-oriented User Requirements Notation (*AoURN*) [47], Activity Theory (*AT*) [48], The Cloud Component Approach (*CCA*), Model Driven Service Engineering (*MDSE*) [49], Object-oriented Software Product Line Modelling (*OO-SPL*) [50], and Reusable Aspect Models (*RAM*) [51, 52]. The comparison criteria were grouped into two broad categories—*Modelling Dimensions* and *Key Concepts*. Modelling dimensions include properties like Phase, Notation, and Units of Encapsulation. Key concepts, on the other hand, provide an insight into parameters like Paradigm, Modularity, Composability, Traceability and Trade-off Analysis. Of these six approaches, *AoURN* [47, 53] and *OO-SPL* [50] are of interest to this work, as both these approaches are applicable in the Early and Late Requirements phases of software development. The i* modelling notation belongs to this approach. In fact, *AoURN* is based on the ITU-T Z.151 [54] standard that uses Goal-oriented Requirements Language (GRL), that is based on i* modelling. *AoURN* is machine analysable and can perform scenario regression tests, goal-model evaluation, and trade-off analysis. Unlike the other modelling approaches, *AoURN* provides structural, behavioural, and intentional views, along with generic support for qualities and non-functional properties. It is purely graphical in nature.

Most model checking techniques are best suited for the design and subsequent phases of the development life cycle. Architectural [55] and detailed design [56] model checking have been proposed. Automated verification of requirement models requires a completely different set of ontologies and, hence, existing techniques cannot be extended to requirement models. However, work has been done on the application of model checking to requirement models. Heitmeyer et.al. have proposed a tool support for the SCR tabular notation for requirements specification [57, 58] that supports formal techniques for consistency and completeness checking, model checking, theorem proving and animation. The RSML language [59] has better structuring mechanisms than the SCR notation and also provides a tool support for completeness and consistency checking. Both these approaches are restricted to the domain of embedded systems and process control. Neither of these tools support the goal ontologies that have been proposed for early requirements engineering. Wang has explored the application of ConGolog [60] formalisms to the i* framework for analysing early requirement specification. The work tries to map i* SR diagram concepts to ConGolog primitives and control structures. ConGolog is based on *situation*

calculus [61] and provides a formal machinery for proving assertions on requirement specifications.

Telos [62] captures the i* meta-framework which describes all the semantics and constraints of the i* framework. Telos is equipped with the ability to perform different types of analysis and also check the consistency between i* models. Tropos [63] is an agent-oriented system development framework that utilizes the i* modelling framework to model agent requirements and system configurations. Formal Tropos [3, 64] associates the Tropos methodology to a formal specification language. Formal Tropos allows the specification of constraints, invariants, pre- and post- conditions, thereby capturing the semantics of the i* graphical models. These models can be validated using model checking.

Research Gap

Performing a model transformation requires a clear understanding of the abstract syntax and semantics of both the source and target. Most model-driven engineering practises offer a black box view of the transformation logic making it difficult to observe the operational semantics of a transformation. In order to perform model checking on any sequence-agnostic goal model (like i*), we must first transform the model into some form of finite state model that provides a possible sequencing of activities within the enterprise. This research direction has been explored in Chapter 4.

2.4 Semantic Annotations of Goal Models

The domain of goal model maintenance requires analysts to explore the space of goal model configurations that can be derived from a given erroneous goal model. However, the research question becomes even more complex if we try to go beyond the structural features and consider the semantics of goal models. The nomenclature of goal models do not capture the semantics of the goals. There has been very limited work in the existing literature that annotates goal models with more information. This section highlights some of the research that has been done in the domain of annotating requirement models with different types of attributes.

Annotation of Goal Models

Liaskos and Mylopoulos [65] have identified the sequence agnostic nature of standard goal modelling notations like i* [2, 14] and annotated them with temporal logistics for deriving AI-based goal satisfaction planning. The authors introduce the notion of *precedence links* and *effect links* that annotate the i* model with preconditions and postconditions of fulfilling a goal. This kind of ordering allows formalization of goal models using temporal logics (like LTL, CTL, etc.). Although this method establishes some sort of a sequence between the tasks of a goal model, the notion of precedence does not remain intuitive for softgoals. Softgoal satisfaction can be facilitated with *hurt* and *help* contributions from tasks, hard goals, etc.

In [66], Liaskos et al. have highlighted the importance of augmenting goal models (like i*) with the optional requirements or preferences of the users. This paper uses the *precedence* and *effect* links proposed in [65]. Additionally, this work introduces the notion of weighted contribution links for evaluating the degree of satisfaction or denial for softgoals. Accumulation and propagation of these weighted contributions follow the rules prescribed in [67]. Optional user requirements are defined as *Optional Condition Formulae* (OCFs) using first order satisfaction and domain predicates. Preferences are captured as linear combinations of OCFs and these preference formulae may be weighted or non-weighted in nature. Alternate goal plans are evaluated based on the degree of satisfaction of the preferences.

Koliadis and Ghose have been working with semantic effect annotations of business process models [68–70]. In [68], the authors propose the GoalBPM methodology that maps business process models (using BPMN) to high-level stakeholder goals described using the KAOS framework [4]. This is done by defining two types of links—*traceability* and *satisfaction*. The former links goals to activities while the latter links goals to entire business processes. Satisfaction links require effect annotation of the business process model, followed by identification of a set of critical trajectories and, finally, identifying the subset of traceability links that represent the satisfaction links. In [69, 70], the authors have worked with semantic effect annotation and accumulation over business process models (Process SEER) and how it can be extended to check for business process compliance using the PCTk toolkit.

Kaiya et al. [71] have proposed the popular Attributed Goal-Oriented Requirements Analysis (AGORA) method that derives a goal graph from goal models and annotates the nodes and edges of the graph with attribute values and quality matrices. Attribute values consist of contribution values and preference matrices. Contribution values are used to annotate the edges of the goal graph and represent the contribution of the sub-goal towards the fulfillment of the higher-level goal. Preference matrices are the vertex annotations and represent the preference of respective goals to the concerned stakeholders. Both these attribute annotations can be used by analysts to choose among multiple alternate strategies, to perform conflict management and change management. Quality metrics are used to analyze the quality of the requirement specifications that are derived from the goal graphs. The metrics for measuring such quality may be correctness, unambiguity, completeness, inconsistency, etc. Yamamoto and Saeki [72] have extended the idea of using annotated goal graphs for requirements analysis to software component selection.

Research Gap

Researchers have attempted to annotate goal models with temporal information for simulation and model checking purposes. They have also tried to identify effects of goal fulfillment for evaluating user preferences. Semantic annotation of goal models may seem intuitive and analogous to effect annotation of business process models but it is not so. Goal models and process models have completely different objectives and characteristics. The most crucial differential characteristic being the sequence-agnostic nature of goal models. In this perspective, it becomes necessary to spell out a mechanism for semantic annotation of goal model artefacts, and how these goal

semantics can be reconciled over the entire enterprise for performing goal model maintenance. Chapter 5 proposes a framework for doing this and also provides a roadmap to implement it.

References

1. Lapouchnian A (2005) Goal-oriented requirements engineering: an overview of the current research, Depth Report. University of Toronto, Toronto, Canada
2. Yu E, Modelling strategic relationships for process reengineering. PhD thesis, University of Toronto, Toronto, Canada
3. Fuxman AD (2001) Formal analysis of early requirements specifications. MS thesis, Department of Computer Science, University of Toronto, Canada
4. van Lamsweerde A, Darimont R, Letier E (1998) Managing conflicts in goal-driven requirements engineering. Trans Softw Eng Special Issue Inconsistency Manage Softw Dev 24(11):908–926. https://doi.org/10.1109/32.730542
5. Ross D (1977) Structured analysis (SA): a language for communicating ideas. IEEE Trans Softw Eng 3(1):16–34. https://doi.org/10.1109/TSE.1977.229900
6. Greespan S, Borgida A, Mylopoulos J (1986) A requirements modeling language and its logic. Knowl Based Manage Syst 471–502. https://doi.org/10.1016/0306-4379(86)90020-7
7. Bois PD (1995) The albert ii language: on the design and use of a formal specification language for requirements analysis. PhD thesis, Notre Dame de la Paix, Namur, Belgium
8. Chung L et al, Non-functional requirements in software engineering. Kluwer Academic Publishers. https://doi.org/10.1007/978-1-4615-5269-7. ISBN 978-1-4615-5269-7
9. Mylopoulos J, Chung L, Nixon B, Representing and using non-functional requirements: a process-oriented approach. IEEE Trans Softw Eng 18(6). https://doi.org/10.1109/32.142871
10. Dardenne A, van Lamsweerde A, Fickas S (1993) Goal-directed requirements acquisition. Sci Comput Program 20(1–2):3–50. https://doi.org/10.1016/0167-6423(93)90021-G
11. van Lamsweerde A, Letier E (2002) From object orientation to goal orientation: a paradigm shift for requirements engineering. In: Proceedings of the 9th international workshop on radical innovations of software and systems engineering in the future (Lecture Notes in Computer Science 2941), pp 325–340. https://doi.org/10.1007/978-3-540-24626-8_23
12. Anton A (1996) Goal-based requirements analysis. In: Proceedings of the 2nd IEEE international conference on requirements engineering (ICRE), pp 136–144. https://doi.org/10.1109/ICRE.1996.491438
13. Anton A (1997) Goal identification and refinement in the specification of software-based information systems. PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA
14. Yu E (1997) Towards modeling and reasoning support for early-phase requirements engineering.In: Proceedings of the 3rd international symposium on requirements engineering (RE), pp 226–235. https://doi.org/10.1109/ISRE.1997.566873
15. Abbas S, Seba H (2012) A module-based approach for structural matching of process models. In: Proceedings of the 5th IEEE international conference on service-oriented computing and applications (SOCA), pp 1–8. https://doi.org/10.1109/SOCA.2012.6449441
16. Wang J, Liu H, Wang H (2014) A mapping-based tree similarity algorithm and its application to ontology alignment. Knowl Based Syst 56:97–107. https://doi.org/10.1016/j.knosys.2013.11.002
17. Mao M, Peng Y, Spring M (2010) An adaptive ontology mapping approach with neural network based constraint satisfaction. Web Semant Sci Serv Agents World Wide Web 8(1):14–25. https://doi.org/10.1016/j.websem.2009.11.002
18. Kalyanpur A, Pastor DJ, Battle S, Padget JA (2004) Automatic mapping of OWL ontologies into java. In: Proceedings of the sixteenth international conference on software engineering and knowledge engineering (SEKE), pp 98–103

19. Arnold P, Rahm E (2014) Enriching ontology mappings with semantic relations. Data Knowl Eng 93:1–18. https://doi.org/10.1016/j.datak.2014.07.001
20. Khattak A, Pervez Z, Khan W, Khan A, Latif K, Lee S (2015) Mapping evolution of dynamic web ontologies. Inf Sci 303(C):101–119. https://doi.org/10.1016/j.ins.2014.12.040
21. Wang P, Xu B, Lu J, Kang D, Zhou J (2006) Mapping ontology relations: an approach based on best approximations. In: Proceedings of the 8th Asia-Pacific Web conference on Frontiers of WWW Research and Development, APWeb'06 (Lecture Notes in Computer Science 3841), pp 930–936. https://doi.org/10.1007/11610113_97
22. Kumar SK, Harding JA (2013) Ontology mapping using description logic and bridging axioms. Comput Ind 64(1):19–28. https://doi.org/10.1016/j.compind.2012.09.004
23. Arch-int N, Arch-int S (2013) Semantic ontology mapping for interoperability of learning resource systems using a rule-based reasoning approach. Expert Syst Appl 40(18):7428–7443. https://doi.org/10.1016/j.eswa.2013.07.027
24. Zawawy H, Mankovskii S, Kontogiannis K, Mylopoulos J (2015) Mining software logs for goal-driven root cause analysis. Art Sci Anal Softw Data Chap 18:519–554
25. Ghose A, Santiputri M, Saraswati A, Dam HK (2014) Data-driven requirements modeling: some initial results with i*. In: Tenth Asia-Pacific conference on conceptual modelling (APCCM), vol 154, pp 55–64. http://dl.acm.org/citation.cfm?id=2667691.2667698
26. Jin X, Donze A, Deshmukh JV, Seshia SA (2015) Mining requirements from closed-loop control models. IEEE Trans Comput Aided Des Integr Circ Syst 34(11):1704–1717. https://doi.org/10.1109/TCAD.2015.2421907
27. Qi J, Zhang Z, Jeon S, Zhou Y (2016) Mining customer requirements from online reviews: A product improvement perspective. Inf Manage Elsevier 53(8):951–963. https://doi.org/10.1016/j.im.2016.06.002
28. Garcia JE, Paiva AC (2016) Maintaining requirements using web usage data. Proc Comput Sci 100(Supplement C):626–633. https://doi.org/10.1016/j.procs.2016.09.204
29. Knauss A, Damian D, Franch X, Rook A, Müller HA, Thomo A (2016) ACon: a learning-based approach to deal with uncertainty in contextual requirements at run time. Inf Softw Technol Elsevier 70(Supplment C):85–99. https://doi.org/10.1016/j.infsof.2015.10.001
30. Hassani M, Siccha S, Richter F, Seidl T (2015) Efficient process discovery from event streams using sequential pattern mining. In: IEEE symposium series on computational intelligence, pp 1366–1373. https://doi.org/10.1109/SSCI.2015.195
31. Sohrabi MK, Ghods V (2016) CUSE: a novel cube-based approach for sequential pattern mining. In: 4th international symposium on computational and business intelligence (ISCBI) pp 186–190. https://doi.org/10.1109/ISCBI.2016.7743281
32. Hassani M, Lu Y, Wischnewsky J, Seidl T (2016) A geometric approach for mining sequential patterns in interval-based data streams. In: IEEE international conference on fuzzy systems (FUZZ-IEEE), pp 2128–2135. https://doi.org/10.1109/FUZZ-IEEE.2016.7737954
33. Chaudhari M, Mehta C (2016) Extension of prefix span approach with grc constraints for sequential pattern mining. In: International conference on electrical, electronics, and optimization techniques (ICEEOT), pp 2496–2498. https://doi.org/10.1109/ICEEOT.2016.7755142
34. Fei X, Zheng S, Li-jing Y, Chao F (2016) A improved sequential pattern mining algorithm based on prefixspan. World Autom Congr (WAC) 1–4. https://doi.org/10.1109/WAC.2016.7583059
35. Patel R, Chaudhari T (2016) A review on sequential pattern mining using pattern growth approach. In: International conference on wireless communications, signal processing and networking (WiSPNET), pp 1424–1427. https://doi.org/10.1109/WiSPNET.2016.7566371
36. Zhu X, Wu S, Zou G (2015) User behavior detection for online survey via sequential pattern mining. In: Fifth international conference on instrumentation and measurement, computer, communication and control (IMCCC), pp 493–497. https://doi.org/10.1109/IMCCC.2015.110
37. Uragaki K, Hosaka T, Arahori Y, Kushima M, Yamazaki T, Araki K, Yokota H (2016) Sequential pattern mining on electronic medical records with handling time intervals and the efficacy of medicines. In: IEEE symposium on computers and communication (ISCC), pp 20–25. https://doi.org/10.1109/ISCC.2016.7543708

38. Abbasghorbani S, Tavoli R (2015) Survey on sequential pattern mining algorithms. In: 2nd international conference on knowledge-based engineering and innovation (KBEI), pp 1153–1164. https://doi.org/10.1109/KBEI.2015.7436211
39. Ni Z, Wang S, Li H (2011) Mining organizational structure from workflow logs. In: Proceeding of the international conference on e-education, entertainment and e-management, pp 222–225. https://doi.org/10.1109/ICeEEM.2011.6137791
40. Schöniga S, Cabanillas C, Jablonski S, Mendling J (2016) A framework for efficiently mining the organisational perspective of business processes. Decis Support Syst Elsevier 89(Supplement C):87–97. https://doi.org/10.1016/j.dss.2016.06.012
41. Cleland-Huang J, Settimi R, Zou X, Solc P (2006) The detection and classification of non-functional requirements with application to early aspects In: 14th IEEE international conference requirements engineering, pp 39–48. https://doi.org/10.1109/RE.2006.65
42. Sendall S, Kozaczynski W (2003) Model transformation: The heart and soul of model-driven software development. IEEE Softw 20(5):42–45. https://doi.org/10.1109/MS.2003.1231150
43. Schönböck J et al (2009) Catch me if you can—debugging support for model transformations, MODELS, Workshops (Lecture notes in computer science 6002(2010)), pp 5–20. https://doi.org/10.1007/978-3-642-12261-3_2
44. Shah SMA, Anastasakis K, Bordbar B (2009) From uml to alloy and back again, MODELS, workshops (Lecture notes in computer science 6002(2010)), pp 158–171. https://doi.org/10.1007/978-3-642-12261-3_16
45. Kerzhner AA, Paredis CJJ (2010) Model-based system verification: a formal framework for relating analyses, requirements, and tests, MODELS, workshops (Lecture notes in computer science 6627(2011)), PP 279–292. https://doi.org/10.1007/978-3-642-21210-9_27
46. Mussbacher G et al (2011) Comparing six modelling approaches, MODELS, workshops (Lecture notes in computer science 7167(2012)), PP 217–243. https://doi.org/10.1007/978-3-642-29645-1_22
47. Mussbacher G (2010) Aspect-oriented user requirements notation. PhD thesis, School of Information Technology and Engineering, University of Ottawa, Canada
48. Georg G (2011) Activity theory and its applications in software engineering and technology. Technical Report CS-11-101, Colorado State University
49. Kathayat SB, Le HN, Bræk R (2011) A model-driven framework for component-based development. In: 15th International SDL Forum Toulouse Integrating System and Software Modeling 2011, France, pp 154–167. https://doi.org/10.1007/978-3-642-25264-8_13
50. Capozucca A, Cheng B, Guelfi N, Istoan P (2011) bcms-oom-spl, repository for model driven development. http://www.cs.colostate.edu/content/bcms-oom-spl
51. Klein J, Kienzle J (2007) Reusable aspect models, 11th workshop on aspect-oriented modelling. Nashville, TN, USA
52. Kienzle J et al (2010) Aspect-oriented design with reusable aspect models, transactions on aspect-oriented software development VII (Lecture notes in computer science 6210), pp 272–320. https://doi.org/10.1007/978-3-642-16086-8_8
53. Mussbacher G, Amyot D, Araújo J, Moreira A (2010) Requirements modelling with the aspect-oriented user requirements notation (AoURN): a case study. In; Transactions on aspect-oriented software development VII: a common case study for aspect-oriented modeling, pp 23–68. https://doi.org/10.1007/978-3-642-16086-8_2
54. User Requirements Notation (URN)—Language Definition (2008) ITU-T: Recommendation Z.151. Geneva, Switzerland. http://www.itu.int/rec/T-REC-Z.151/en
55. Allen R, Garlan D (1994) Formalizing architectural connection. In: Proceedings of the 16th international conference on software engineering, pp 71–80. http://dl.acm.org/citation.cfm?id=257734.257745
56. Cimatti A et al (1998) Formal verification of a railway interlocking system using model checking. J Formal Aspects Comput 10:361–380. https://doi.org/10.1007/s001650050022
57. Heitmeyer C, Jeffords R, Labaw B (1996) Automated consistency checking of requirements specifications. ACM Trans Softw Eng Methodol 5(3):231–261. https://doi.org/10.1145/234426.234431

58. Heninger K (1980) Specifying software requirements for complex system: new techniques and their application. IEEE Trans Softw Eng 6(1):2–13. https://doi.org/10.1109/TSE.1980.230208
59. Levenson N, Heimdahl M, Hildreth H (1994) Requirements specification for process control systems. IEEE Trans Softw Eng 20(9):684–706. https://doi.org/10.1109/32.317428
60. Giacomo GD, Lesperance Y, Levesque H (2000) Congolog, a concurrent programming language based on the situation calculus. J Artif Intell 121(1–2):109–169. https://doi.org/10.1016/S0004-3702(00)00031-X
61. McCarthy J, Hayes P (1969) Some philosophical problems from the standpoint of artificial intelligence. Mach Intell 4:463–504. https://doi.org/10.1016/B978-0-934613-03-3.50033-7
62. Mylopoulos J et al (1990) Telos: representing knowledge about information systems. ACM Trans Inf Syst 8(4):325–362. https://doi.org/10.1145/102675.102676
63. Castro J, Kolp M, Mylopoulos J (2002) Towards requirements-driven information systems engineering: the tropos project. Inf Syst 27(6):365–389. https://doi.org/10.1016/S0306-4379(02)00012-1
64. Fuxman A, Pistore M, Mylopoulos J, Traverso P (2001) Model checking early requirements specifications in tropos. In: Proceedings of the 5th international symposium on requirements engineering (RE), pp 174–181. https://doi.org/10.1109/ISRE.2001.948557
65. Liaskos S, Mylopoulos J (2010) On temporally annotating goal models. In: Proceedings of the 4th international i* workshop—iStar10, pp 62–66
66. Liaskos S, McIlraith SA, Mylopoulos J (2009) Towards augmenting requirements models with preferences. In: IEEE/ACM international conference on automated software engineering, pp 565–569. https://doi.org/10.1109/ASE.2009.91
67. Sebastiani R, Giorgini P, Mylopoulos J (2004) Simple and minimum-cost statisfiability for goal models. In: Proceedings of the 16th international conference on advanced information systems engineering (CAiSE'04), pp 20–35. https://doi.org/10.1007/978-3-540-25975-6_4
68. Koliadis G, Ghose A (2006) Relating business process models to goal-oriented requirements models in KAOS. In: Advances in knowledge acquisition and management, pacific rim knowledge acquisition workshop (PKAW), pp 25–39. https://doi.org/10.1007/11961239_3
69. Hinge K, Ghose A, Koliadis G (2009) Process SEER: a tool for semantic effect annotation of business process models. In: Proceedings of the 13th IEEE international conference on enterprise distributed object computing (EDOC), pp 54–63. https://doi.org/10.1109/EDOC.2009.24
70. Ghose A, Koliadis G (2008) PCTk: a toolkit for managing business process compliance. In: Proceedings of the 2nd international workshop on Juris-Informatics (JURISIN'08)
71. Kaiya H, Horai H, Saeki M (2002) AGORA: attributed goal-oriented requirements analysis method. In: Proceedings of the IEEE joint international conference on requirements engineering, pp 13–22. https://doi.org/10.1109/ICRE.2002.1048501
72. Yamamoto K, Saeki M (2007) Using attributed goal graphs for software component selection: an application of goal-oriented analysis to decision making. In: 26th international conference on conceptual modeling ER '07 tutorials, posters, panels and industrial contributions, vol 83, pp 215–220. http://dl.acm.org/citation.cfm?id=1386957.1386992