A Conceptual Framework of Research on Visual Language Specification Languages

Anitta Thomas
School of Computing
University of South Africa
Florida Park, South Africa
thomaa@unisa.ac.za

Aurona J. Gerber
Department of Informatics/CAIR
University of Pretoria
Pretoria, South Africa
aurona.gerber@up.ac.za

Alta van der Merwe
Department of Informatics
University of Pretoria
Pretoria, South Africa
alta.vdm@up.ac.za

***Abstract*—Visual languages make use of spatial arrangements of graphical and textual elements to represent information. Domain specific diagrams, including flow charts and music sheets, are examples of visual languages. An established area of research is the study of languages which can be used to create declarative specifications of visual languages. In this paper, the result of a review of research on visual language specification languages is presented. Specifically, a structured literature review is conducted to establish research themes by analysing what has been studied in the context of specification languages. The result of the literature review is used to develop a conceptual framework that consists of six research themes with related topics. Additionally, discussions on how the conceptual framework can be used as a basis to guide research in the field of specification languages, to perform feature based characterisations and to create lists of criteria to evaluate and compare specification languages are included in this paper.**

***Index Terms*—visual languages, diagrams, specifications, specification languages**

I. Introduction

In computing, languages that make use of non-sequential arrangements of graphical and textual elements are studied as visual languages. Each visual language has its own rules about the set of elements and how these elements should be spatially arranged to form valid instances of the language [1]. Examples of visual languages are flow charts, finite state automata diagrams and music sheets, where the former two and the latter one are used in the domains of computing and music respectively.

An area of study is the generation and utilisation of formal specifications of visual languages, where the specifications are generated using appropriate languages, referred to as specification languages. Visual language specifications are explicit descriptions, in general, of either the syntax or semantics of the language [1]. Formal declarative specifications are beneficial to facilitate automated processing of visual languages [2]. Automated processing is generally used to realise applications to interpret or generate visual languages [3].

The automated processing of visual languages can be used to provide software solutions within a variety of contexts. For example, automated interpretation can be used to generate textual summaries, which improve accessibility of diagrams for visually impaired users [4]. Automated processing can also be useful, for example, to generate annotations that describe the content of images of diagrams for the realisation of the semantic web [5].

Numerous specification languages with different underlying formalisms have been developed for visual language specifications [1]. Examples of specification languages are modified string grammars [6], graph grammars [7] and the Web Ontology Language [8]. Research on specification languages has a long history (see the survey in [1], for example), and includes studies on algorithms for processing specifications [9] [10], properties of these algorithms [11] [12], software applications that use specifications [7] [13] and tools for generating specifications [11] [14]. Points of focus for research in this field vary from the theoretical to the practical. Establishing time complexity theoretically [15] [16] as opposed to time usage empirically [17] [18] is one example of different points of focuses of studies in specification languages.

The aim of this paper is to present an overview of the current research on specification languages, by amalgamating various studies in this field. This overview is developed through a literature review and presented as a conceptual framework of research in the field. Specifically, a structured literature review is conducted to explore aspects that have been researched in this area. It should be emphasised that the aim of the review is not to establish the list of different specification languages similar to the review in [1] for example, but rather to present a unified view of research topics in the field.

It is envisaged that the developed framework can be used as a basis to guide further research in the field. For example, the framework may be used as a basis to create comprehensive lists of features that can be used to characterise or evaluate or compare specification languages. Similarly, the framework may be used to guide research into new specification languages.

This paper is structured into five sections. Section II describes the details of the structured literature review. The conceptual framework developed from the literature review is discussed in Section III. Section IV includes discussions on how the conceptual framework can be used for different purposes. A conclusion to the paper is included in Section V.

II. Structured Literature Review

Structured literature review is a research method by which the collection and analysis of relevant literature is performed with specific goals [19]. This method makes it explicit how literature is gathered and analysed [20] to achieve transparency and repeatability [19].

In a 2017 article in the Journal of Visual Languages & Computing, a structured analysis of the literature was used to provide a comprehensive definition of visual languages [21]. As this research falls in the realm of visual languages, the structured review used in this research was designed similarly to the study in [21]. The structured review in this research is guided by two questions; what aspects and how they have been studied in the field of visual language specification languages.

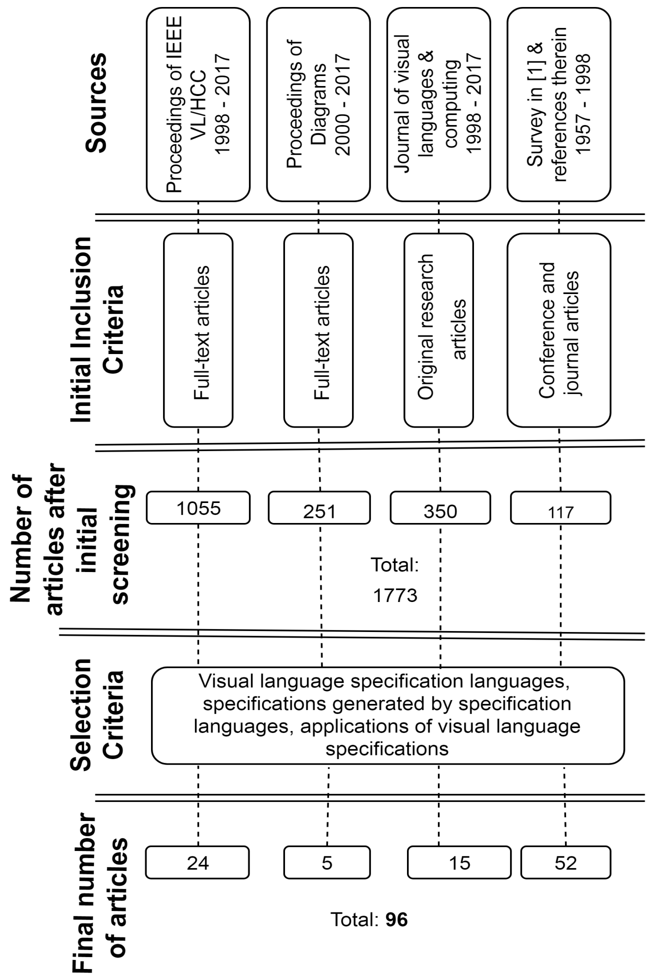
Similar to [21], the selected sources for the present literature review were chosen to be the proceedings of the IEEE Visual Languages (VL)/Human Centric Computing (HCC) and Diagrams conferences, Journal of Visual Languages & Computing (JVLC), the survey article about specification languages [1], and the references therein. Our choice of these sources is motivated by the fact that the selected conferences and journal are considered to be the most prominent avenues for publications in visual languages [21]. Additionally, the work presented in [1] is acknowledged as a comprehensive survey at the time of its publication [9] [22].

Our next step was to select the period of publications of the selected sources. Since the comprehensive survey in [1] was published in 1998, we assumed that it contains references to most, if not all, of the relevant work up until 1998. The article in [1] contains references published between 1957 and 1998. The first Diagrams conference was held in 2000, so the proceedings of this conference is only available from 2000. It was decided, therefore, that the proceedings of IEEE VL/HCC and JVLC between 1998 and 2017, the proceedings of Diagrams between 2000 and 2017, the survey article in [1] and all the references included in [1] would be considered for the review.

A set of initial screening criteria was applied to the articles in the selected avenues and periods of publications. For proceedings of both conferences, all publications, except those that only had abstracts, were included for the review. Only abstracts were typically included for keynote addresses, panel discussions and workshops. In case of the journal articles, all articles that were marked as original research articles were included for the review. The survey article in [1] and the references therein that were published in conference proceedings and journals were also included. The initial set, after applying the initial screening criteria, consisted of 1773 articles.

The next step was to choose relevant articles from the initial set of 1773 articles. The set of criteria applied was whether an article reported on visual language specification languages, visual language specifications or applications of specifications. This set of criteria was applied to each article by reading the abstracts in detail and referring to the articles where necessary. The total number of articles, after applying the set of inclusion criteria, was 96[1], which is the number of articles used for detailed analysis.

Fig.1 depicts the sources, periods, number of articles and selection criteria used for the review. The number of articles indicated for the publication source [1], includes [1] and the references therein. This number, therefore, indicates the references used as opposed to where they are published. In fact, numerous of these references were published in the proceedings of IEEE VL/HCC conferences and JVLC, prior to 1998.



Sources
Proceedings of IEEE VL/HCC 1998 - 2017
Proceedings of Diagrams 2000 - 2017
Journal of visual languages & computing 1998 - 2017
Survey in [1] & references therein 1957 - 1998
Initial Inclusion Criteria
Full-text articles
Full-text articles
Original research articles
Conference and journal articles
Number of articles after initial screening
1055
251
350
117
Total: 1773
Selection Criteria
Visual language specification languages, specifications generated by specification languages, applications of visual language specifications
Final number of articles
24
5
15
52
Total: 96

Fig. 1. Sources, periods, number of articles and selection criteria used in this literature review

Each article in the set of 96 articles was read in detail by the first author. While reading each article, aspects studied in the context of visual language specification techniques, and how they were studied, were noted down. All the noted aspects were then analysed and grouped into themes, and these themes were used to construct the conceptual framework of research.

III. Conceptual Framework of Research

The result of the literature review was used to develop a conceptual framework of research in the field of visual

[1]The complete list of articles is available at https://bit.ly/2Sv1xgW.

language specification languages. The framework consists of six research themes with related topics in these themes, which are presented in the following sub sections.

Due to space limitations only a subset of articles used for the review, from which the conceptual framework was developed, is included in this discussion.

A. Visual Language Models

In this theme, modelling of visual languages is studied. Modelling refers to how relevant aspects of a visual language, syntactical structure for example, are conceptually modelled and specified in a given specification language. Different specification languages can support different visual language models, for example [10] and [23] uses two different visual language models.

There are different ways to describe visual language models. A visual language specification can be seen as a representation of the visual language model applied to a specific visual language [24] [25]. Formal models can also be used to specify models [26] [27]. Informal textual descriptions can also be used to describe visual language models [24] [25].

B. Specifications

In this theme, topics directly related to the specifications generated by a specification language are included. Two main topics in this theme are the generation of specifications for relevant visual languages and investigations of properties of the generated specifications.

The first aspect in this theme is the development of specifications of visual languages of interest [8] [28]. The specifications are instantiations of visual language models for the chosen visual languages. Specifications can also be used to demonstrate *expressiveness*, a property (see section III-C), of the specification language [29] [30].

The second aspect in this theme is whether a specification satisfies certain properties. Examples of properties studied are *correctness* [31] and *completeness* [6]. Certain properties are specifically studied for grammar based languages, *parsability* [31] [32] is one such property. Properties can be established, for example, using proofs [6] [31] and conditions to confirm properties in specifications [32].

C. Specification Languages

In this theme, properties and artefacts of specification languages are studied. The term artefacts is used to refer to algorithms or techniques used for processing specifications independent of possible implementations, i.e. computer programs.

Two computational properties widely studied are *expressiveness* [11] [12] [25] [27] and *computational costs* [16] [33]. Explanations using examples are often used to demonstrate types or aspects of visual languages that can be described in a given specification technique [29] [30]. Computational costs are measured using time and space complexity [15] [25]. Time complexity can be studied theoretically [15] [16] or empirically [17], and in some cases theoretical time complexity is compared to practical cases [18]. Practical cases involve measuring the processing time taken by a software application that uses visual language specifications [18].

In general, highly expressive specification languages are desirable [11] [25] [30], however, high expressiveness often leads to high computational costs [1] [9]. In the context of specification languages, there is an emphasis on ways to improve computational efficiency either by limiting expressiveness of the specification language [33] [34] or by exploring different ways of processing specifications [35] [36].

The formal property *decidability* has been studied for different specification languages with different underlying formalisms [23] [37]. Properties such as *soundness* and *completeness* are primarily studied in the context of logic-based specification languages [3] [23] [24]. When the above-mentioned properties are studied for specification languages, they are often presented formally with proofs [3] [37]. In [30], *generality* of a specification language is demonstrated by comparing the expressiveness of different specification languages to that discussed in the article using proof outlines.

When it comes to artefacts, a noticeable trend in logic based specification languages is the reuse of existing artefacts while in grammar based languages is to develop or modify an existing artefact for processing visual language specifications. The most frequently developed and studied artefact in grammar based languages is the parsing algorithm [9] [10] [16] [25] [27] [32] [33]. Some researchers have explored ways to use parsing algorithms to check for possible run-time errors in a given grammar during parsing [38] and to suggest ways to correct an input diagram for successful parsing [10] [39]. Similarly, in [36], an algorithm is presented that can be used during parsing to correct errors in an input diagram. There are also studies that focus on different properties of parsing algorithms. Examples of properties of parsing algorithms are *correctness* [40] and *termination* [25].

Another class of artefacts is techniques to automatically generate parsers given a visual language specification generated by a grammar based language [41] [42]. Techniques for automated parser generation are aimed at reducing the time and effort required to generate visual language specific tools. For example, an algorithm for automated generation of parsers is given in [41].

D. Classification of Visual Languages

In this theme, how the expressiveness of the specification language can be used to classify visual languages is studied. For example, in [43] [44] the classes of visual languages that can be defined using different forms of grammars are presented. Results of studies in this theme are schemes used for classifications and classifications of visual languages using the developed schemes.

E. Software Applications

In this theme, how the visual language specification generated by a specification language can be used in software applications is studied. In general, the range of software applications is significantly varied in terms of their input and output, and generalisability of these applications, i.e. designed to process one or multiple visual languages.

The first category of software applications is dedicated to interpret a visual language. Examples of studies that report on the first category of applications are [17] [45]. The second category is dedicated diagram editors that can check syntactic validity of diagrams, either while the diagram is being drawn or after completing the diagram by the user. These dedicated diagram editors can also assist with the completion of diagrams [13], indicate errors in the diagram [39] [46] as well as suggest possible solutions to solve the errors in the diagram [39]. The third category of applications automatically generates software components for visual language processing [10] [12]. These software components are typically parsers and visual language editors for processing visual languages. Unlike the first three categories, the fourth category of applications processes visual languages beyond syntactical level, typically to support processing of visual languages at the semantic level, for example to process models represented in diagrams [7].

All four categories of software applications use specifications of relevant visual languages as input. Additionally all categories except third take diagrams as input and produce the result of interpretation of diagrams as output. A variety of visual language input and output formats have been used in these applications. Examples of input formats are hand-drawn [27], computer-generated [36] and scanned [47] diagrams. The output formats are often application dependent [25] [27].

Software applications are often presented using their design, implementation and evaluation. Details of the application including the general architecture of the system [48], various components and their functionality [10] and the processing steps [49] are generally discussed. The details of the application be it the general design or functionality vary from one publication to another. Sometimes the implementation language [7] [36] and the details of the operating system on which the application is meant to be executed [18] are also included.

The most common form of evaluation is empirical evaluation. When presenting the empirical evaluation, mostly the number of diagrams and measurements of various features of the application are indicated. The most common features measured are time taken to process [7] [17] [27] [36] and accuracy of interpretation [50] [51] of diagrams. Sometimes for the third category of applications, time taken to generate software tools [52] and the length of code of the generated tool [18] are noted. The details of the system on which evaluations are conducted are also listed [36].

F. Tools

In this theme, how tools can be realised to develop visual language specifications are studied.

A number of tools have been developed to support the development of visual language specifications, and they include dedicated editors, often graphical [11] [14] [32], with the ability to specify visual languages visually [11] [31], to check and debug specifications [11] [42] and to visualise aspects of specifications [42]. Similar to software applications (see section III-E), software tools can also be presented using their design, implementation and evaluation [11].

G. Summary and Discussion

A summary of the conceptual framework is illustrated in Fig.2.

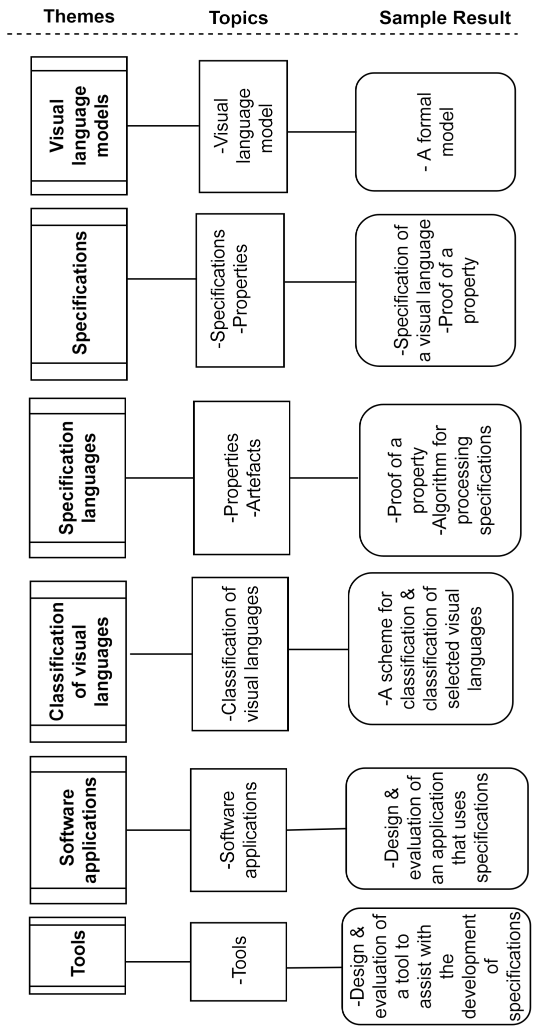

Themes
Topics
Sample Result
Visual language models
-Visual language model
- A formal model
Specifications
-Specifications
-Properties
-Specification of a visual language
-Proof of a property
Specification languages
-Properties
-Artefacts
-Proof of a property
-Algorithm for processing specifications
Classification of visual languages
-Classification of visual languages
-A scheme for classification & classification of selected visual languages
Software applications
-Software applications
-Design & evaluation of an application that uses specifications
Tools
-Tools
-Design & evaluation of a tool to assist with the development of specifications

Fig. 2. A concise depiction of the conceptual framework of research on visual language specification languages

To summarise, all six themes of the framework are focussed around visual language specification languages. The first theme, *visual language models*, focuses on the visual language models supported by the specification language. The second theme, *specifications*, centers around the development of specifications for visual languages and their properties. Properties and artefacts of a specification language are the main topics of research in the third theme, *specification languages*. The theme, *classification of visual languages*, focuses on classifying visual languages using properties of the selected specification language. Utilisation of specifications generated by the specification language is studied in the theme of *software applications*. The theme *tools* focuses on tools to assist in the development of specifications. Although tools are also software applications, in this framework software that makes use of specifications is included in the theme *software applications*, while software that is used to support the development of specifications is included in the theme *tools*.

The conceptual framework presents an overview of what has been studied in the field. However, there are aspects that have been mentioned as relevant but not studied in the field. For example, industry relevance [1] and ease of use [7] [11] [31] can be relevant features of a specification language. Therefore, investigations into industry relevance and ease of use of a specification language, for example, can be beneficial in the field. Inclusion of such relevant aspects can possibly improve the conceptual framework.

IV. Potential Uses of the Conceptual Framework

The conceptual framework presented in Section III can be used to guide research in the field of visual language specification languages. In this section, a few possible ways to utilise the framework are briefly discussed.

Firstly, when a new specification language is studied, the framework can be used as a basis to carry out research into the language. Research into a new specification language can therefore be planned according to the topics in the six themes. One possible order of topics to study for a new specification language is the order in which the themes are presented in Section III. Investigations into certain topics are prerequisites for other topics of investigations. For example, a software application (topic in the theme *software applications*) can only be realised if a specification of the relevant visual language can be generated (topic in the theme *specifications*) using the specification language.

Secondly, the framework may be used to identify gaps in knowledge about an existing specification language. For example, a literature review can be conducted into the topics listed under each research theme to identify what is studied and can be studied further for a given specification language.

Thirdly, the developed framework provides a basis for lists of aspects that can be used for wider characterisations of specification languages. Although aspects such as underlying formalisms and computational costs [1] are the well-known features to characterise, the topics listed under each theme of the framework can be used to develop a detailed list of features to characterise specification languages. For example, the visual language models supported (topic in the theme *visual language models*), the classes of visual languages that can be specified (topic in the theme *specification techniques*), specification language properties (topic in the theme *specification techniques*) and tool support (based on the theme *tools*) can be used to characterise a specification language.

Fourthly, the topics in the conceptual framework can also be used as a basis to develop evaluation criteria to evaluate and compare specification languages. For example, availability of tools (based on the theme *tools*) and the classes of visual languages that can be expressed (topic in the theme *specification techniques*) are two features that can be used to evaluate and compare different specification languages.

V. Conclusion

In this paper, a conceptual framework of research on visual language specification languages is presented. The framework is developed using a structured literature review and consists of six themes with related topics in each theme. Brief discussions of different ways in which the framework can be used as a basis to guide further research in the field are also included.

The conceptual framework is essentially an overview of research in the field of visual language specification languages. The framework can be expanded by including themes and topics that are mentioned as relevant in the literature but not necessarily studied in the field. Similarly, including more sources of literature in the review may result in the identification of further themes and topics that can be added to the conceptual framework.

The developed framework is currently being used to design investigations into a new specification language. The framework provides a solid outline to study new specification languages and at the same time offers flexibility in terms of details of investigations for a given context.

References

[1] K. Marriott, B. Meyer, and K. B. Wittenburg, "A Survey of Visual Language Specification and Recognition," in *Visual Language Theory*, K. Marriott, B. Meyer, eds. New York, NY, USA: Springer-Verlag, 1998. pp. 5–85.

[2] R. A. Kirsch, "Computer Interpretation of English Text and Picture Patterns," *IEEE Transactions on Electronic Computers*, vol. EC-13, pp. 363–376, Aug. 1964.

[3] R. Helm and K. Marriott, "A declarative specification and semantics for visual languages," *Journal of Visual Languages & Computing*, vol. 2, no. 4, pp. 311–331, 1991.

[4] O. T. Babalola, "Automatic Recognition and Interpretation of Finite State Automata Diagrams," Master's thesis, Department of Mathematical Sciences (Computer Science), University of Stellenbosch, 2015.

[5] L. Yu, *A Developer's Guide to the Semantic Web.* Berlin Heidelberg: Springer-Verlag, 2011.

[6] A. C. Shaw, "A Formal Picture Description Scheme as a Basis for Picture Processing Systems," *Information and Control*, vol. 14, no. 1, pp. 9–52, 1969.

[7] L. Grunske, K. Winter, and N. Yatapanage, "Defining the Abstract Syntax of Visual Languages with Advanced Graph Grammars: A Case Study based on Behavior Trees," *Journal of Visual Languages & Computing*, vol. 19, no. 3, pp. 343–379, 2008.

[8] A. Thomas, A. J. Gerber, and A. van der Merwe, "An Investigation into OWL for Concrete Syntax Specification Using UML Notations," in *Diagrammatic Representation and Inference: Proc. of 9^{th} Int. Conf., Diagrams 2016, Philadelphia, PA, USA, August 7-10, 2016*, M. Jamnik, Y. Uesaka, S. E. Schwartz, eds. Switzerland: Springer, 2016. pp. 197–211.

[9] G. Costagliola and G. Polese, "Extended Positional Grammars," in *Proceedings of the 2000 IEEE International Symposium on Visual Languages*, 2000. pp. 103–110.

[10] G. Costagliola, V. Deufemia, G. Polese, and M. Risi, "Building syntax-aware editors for visual languages," *Journal of Visual Languages & Computing*, vol. 16, no. 6, pp. 508–540, 2005. Selected papers from Visual Languages and Formal Methods 2004 (VLFM '04).

[11] G. Costagliola, M. D. Rosa, and V. Fuccella, "Extending local context-based specifications of visual languages," *Journal of Visual Languages & Computing*, vol. 31, pp. 184–195, 2015. Special Issue on DMS2015.

[12] G. Toffetti and M. Pezzè, "Graph transformations and software engineering: Success stories and lost chances," *Journal of Visual Languages & Computing*, vol. 24, no. 3, pp. 207–217, 2013. Part Special Issue on Graph Transformation and Visual Modeling Techniques.

[13] S. Mazanek, S. Maier, and M. Minas, "Auto-completion for diagram editors based on graph grammars," in *Proceedings of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing*, 2008. pp. 242–245.

[14] V. Rafe, M. Golparian, and S. Rasoolzadeh, "Using graph transformation systems to formalize Tropos diagrams," *Journal of Visual Languages & Computing*, vol. 30, pp. 1–16, 2015.

[15] X. Zeng, K. Zhang, J. Kong, and G.-L. Song, "RGG+: An Enhancement to the Reserved Graph Grammar Formalism," in *Proceedings of 2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, 2005. pp. 272–274.

[16] K. J. Peng, T. Yamamoto, and Y. Aoki, "A new parsing scheme for plex grammars," *Pattern Recognition*, vol. 23, no. 3, pp. 393–402, 1990.

[17] R. H. Anderson, "Syntax-directed Recognition of Hand-printed Two-dimensional Mathematics," *Symposium on Interactive Systems for Experimental Applied Mathematics: Proc. of the Association for Computing Machinery Inc. Symposium, New York, NY, USA*, ACM, 1967. pp. 436–459.

[18] E. J. Golin and T. Magliery, "A compiler generator for visual languages," *Proceedings of the 1993 IEEE Symposium on Visual Languages*, 1993. pp. 314–321.

[19] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," tech. rep., Keele University and University of Durham, 2007.

[20] A. Kofod-Petersen, "How to do a Structured Literature Review in computer science," Aug. 2014. [online]. Available: http://www-users.cselabs.umn.edu/classes/Fall-2016/csci8001/papers/reading/lit-review2.pdf. [Accessed: Mar. 20 2019].

[21] M. Erwig, K. Smeltzer, and X. Wang, "What is a visual language?," *Journal of Visual Languages & Computing*, vol. 38, pp. 9–17, 2017.

[22] R. P. Futrelle, "Ambiguity in visual language theory and its role in diagram parsing," in *Proceedings of the 1999 IEEE Symposium on Visual Languages*, 1999. pp. 172–175.

[23] V. Haarslev, "A Logic-based Formalism for Reasoning about Visual Representations," *Journal of Visual Languages & Computing*, vol. 10, no. 4, pp. 421–445, 1999.

[24] P. Bottoni, B. Meyer, and F. Parisi-Presicce, "On a uniform logical framework for diagrammatic reasoning," in *Proceedings of IEEE Symposium on Human-Centric Computing Languages and Environments*, 2001. pp. 64–71.

[25] J. Rekers and A. Schürr, "A Graph Grammar Approach to Graphical Parsing," *Proceedings of the 11th IEEE International Symposium on Visual Languages*, 1995. pp. 195–202.

[26] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer, "Formal integration of inheritance with typed attributed graph transformation for efficient vl definition and model manipulation," in *Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, 2005. pp. 71–78.

[27] S.-K. Chang, "A Method for the Structural Analysis of Two-dimensional Mathematical Expressions," *Information Sciences*, vol. 2, pp. 253–272, July 1970.

[28] F. Drewes and R. Klempien-Hinrichs, "Picking Knots from Trees: The Syntatic Structure of Celtic Knotwork," in *Theory and Application of Diagrams, Proc. of First Int. Conf., Diagrams 2000*, M. Anderson, P. Cheng, V. Haarslev, eds. Springer, 2000. pp. 89–104.

[29] B. Meyer, "Pictures Depicting Pictures on the Specification of Visual Languages by Visual Grammars," in *Proceedings of the 1992 IEEE Workshop on Visual Languages*, 1992. pp. 41–47.

[30] M. A. Najork and S. M. Kaplan, "Specifying visual languages with conditional set rewrite systems," in *Proceedings of the 1993 IEEE Symposium on Visual Languages*, 1993. pp. 12–18.

[31] G. Costagliola, V. Deufemia, and G. Polese, "Visual language implementation through standard compiler-compiler techniques," *Journal of Visual Languages & Computing*, vol. 18, no. 2, pp. 165–226, 2007. Selected papers from Visual Languages and Computing 2005 (VLC '05).

[32] Y. Adachi and Y. Nakajima, "A context-sensitive NCE graph grammar and its parsability," in *Proceeding of the 2000 IEEE International Symposium on Visual Languages*, 2000. pp. 111–118.

[33] M. Tucci, G. Vitiello, and G. Costagliola, "Parsing Nonlinear Languages," *IEEE Transactions on Software Engineering*, vol. 20, pp. 720–739, Sep 1994.

[34] C. Crimi, A. Guercio, G. Nota, G. Pacini, G. Tortora, and M. Tucci, "Relation Grammars for Modelling Multi-dimensional Structures," in *Proceedings of the 1990 IEEE Workshop on Visual Languages*, 1990. pp. 168–173.

[35] H. Bunke, "Attributed Programmed Graph Grammars and Their Application to Schematic Diagram Interpretation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-4, pp. 574–582, Nov 1982.

[36] S. S. Chok and K. Marriott, "Automatic Construction of User Interfaces from Constraint Multiset Grammars," in *Proceedings of the 11th IEEE International Symposium on Visual Languages*, 1995. pp. 242–249.

[37] G. Costagliola and F. Ferrucci, "Symbolic Picture Languages and their Decidability and Complexity Properties," *Journal of Visual Languages & Computing*, vol. 10, no. 4, pp. 381–419, 1999.

[38] G. Costagliola, V. Deufemia, F. Ferrucci, and C. Gravino, "On the pLR Parsability of Visual Languages," in *Proceedings of the 2001 IEEE Symposia on Human-Centric Computing Languages and Environments*, 2001. pp. 48–51.

[39] A.-P. Tuovinen, "Practical Error Handling in Parsing Visual Languages," *Journal of Visual Languages & Computing*, vol. 11, no. 5, pp. 505–528, 2000.

[40] J. J. Pfeiffer, "Parsing graphs representing two dimensional figures," in *Proceedings of IEEE Workshop on Visual Languages*, 1992. pp. 200–206.

[41] G. Costagliola, S. Orefice, G. Polese, G. Tortora, and M. Tucci, "Automatic Parser Generation for Pictorial Languages," in *Proceedings of the 1993 IEEE Symposium on Visual Languages*, 1993. pp. 306–313.

[42] R. Helm, K. Marriott, and M. Odersky, "Building Visual Language Parsers," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '91, New York, NY, USA*, ACM, 1991. pp. 105–112.

[43] J. Feder, "Plex Languages," *Information Sciences*, vol. 3, pp. 225–241, July 1971.

[44] B. Meyer, "Logic and structure of space: Towards a visual logic for spatial reasoning," in *Proceedings of International Symposium on Logic Programming*, D. Miller, ed. MIT Press, 1993.

[45] A. Thomas, A. J. Gerber, and A. van der Merwe, "Ontology-Based Spatial Pattern Recognition in Diagrams," in *14th IFIP WG 12.5 International Conference: Proceedings of Artificial Intelligence Applications and Innovations 2018, Rhodes, Greece, May 25-27, 2018*, L. Iliadis, I. Maglogiannis, and V. Plagianakos, eds., Switzerland: Springer, 2018. pp. 61–72.

[46] J. W. Janneck and R. Esser, "A predicate-based approach to defining visual language syntax," in *Proceedings of IEEE Symposium on Human-Centric Computing Languages and Environments*, 2001. pp. 40–47.

[47] J. L. Pfaltz, "Web Grammars and Picture Description," in *Computer Graphics and Image Processing*, vol. 1, no. 2, pp. 193–220, 1972.

[48] G. Costagliola, V. Vincenzo, and M. Risi, "A Multi-layer Parsing Strategy for On-line Recognition of Hand-drawn Diagrams," in *Proceedings of IEEE Visual Languages and Human-Centric Computing*, 2006. pp. 103–110.

[49] A. C. Shaw, "Parsing of Graph-Representable Pictures," *Journal of the ACM*, vol. 17, pp. 453–481, July 1970.

[50] S.-K. Chang, "Picture Processing Grammar and Its Applications," *Information Sciences*, vol. 3, pp. 121–148, Apr. 1971.

[51] K.-S. Fu and B. Bhargava, "Tree Systems for Syntactic Pattern Recognition," *IEEE Transactions on Computers*, vol. C-22, pp. 1087–1099, Dec 1973.

[52] D.-Q. Zhang and K. Zhang, "VisPro: A Visual Language Generation Toolset," in *Proceedings of 1998 IEEE Symposium on Visual Languages*, 1998. pp. 195–202.