How Notations Are Developed: A Proposed Notational Lifecycle

T. G. Green, Noora Fetais

How Notations are Developed: a Proposed Notational Lifecycle

T R G Green[1] and Noora Fetais[2]

[1]Computer Science Department, University of York, UK
thomas.green@cs.york.ac.uk

[2]Computer Science & Engineering Department, University of Qatar, Qatar
n.almarri@qu.edu.qa

Abstract. Notations develop over time. We propose that they characteristically pass through a series of development stages, starting very simple and becoming more complex, reaching a stage of complexity that hinders their usability: then, often, a new higher-level notation is developed that is once again simple, and will perhaps pass through the same development. Notations develop in this way because the way they are used develops. We propose 5 stages; Iconic, Flowering, Formalising, Support Patchwork, and Rebirth. Examples can be found in the development of various software systems, such as programming languages, CSS, and content-management systems; also in other domains, such as circuit diagrams and music notation.

Keywords: Notations, Usability, Cognitive Dimensions Framework, Development of Notational Systems, Diagrams.

1 Introduction

This paper describes the life cycle of notations and how they evolve. Notations that are used to represent data and code evolve, just as do natural languages that are used for human-human communication [1], but natural languages are unplanned, whereas each successive version of a notation is deliberately intended to be an improvement over the previous version – yet by solving some problems, these improvements create new problems; solutions to those problems raise further problems, and so on. That is the *notational life cycle*.

We start with some short examples of notational development, drawn from accounts of the history of dance notation, the history of algebraic notation, and the history of musical notation, and enquire what is known about the motivations that forced the changes over time. We claim that these descriptions of notational development, which we believe to be typical, give little understanding of the reasons for notational development, beyond saying that the notation 'needed to develop': and most importantly, these descriptions show no awareness that other notations, in other fields, follow parallel courses.

What might be meant by saying that a notation 'needs to develop'? Presumably, that it is not doing its job well enough any longer. All the accounts cited present insightful comments about what jobs their notation needs to do, but none of them gives any more generalised account; so we next consider two attempts to describe the features required of notations at a generalised level, focusing on usability in the sense of human-computer interaction. But both these accounts are purely synchronic, describing what is required of a notation at a particular time, but giving no account of the lifecycles of notations, and so these descriptions are of little help to us.

Is it possible, then, to describe the typical lifecycle of a notation? We claim that it is, and we draw on the examples mentioned to postulate five *stages of notational development*. From here we continue with a case study of a particular notation.

1.1 Examples of development of notations

Dance Notation. How do notations originate and develop? Waters and Gibbons [24] are among the few authors to consider several fields, seeking to discover elements in common. They refer to a 'design language', which seems to be an informal private language used by the designer, and a public, more formal 'notation system': "[There is a] mutually supportive relationship between the abstract design language as it

exists in the designer's mind and the public notational system used to express designs." To illustrate their approach, they consider the field of dance and the historical development of dance notation. Like any other notation, dance notation must meet the needs of the various interested parties (dancers, stage designers, lighting crew, director, etc). Early dance notations included the well-known woodcuts of Arbeau's treatise of 1588 on social dancing, *Orchésographie* (Figure 1). These early forms were purely *pictorial* notations.

Fig. 1 a dance movement a Orchésographie

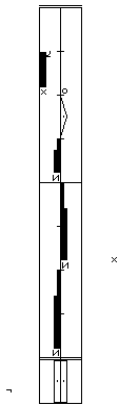

Fig. 2 The basic tango step in Labanotation [26]

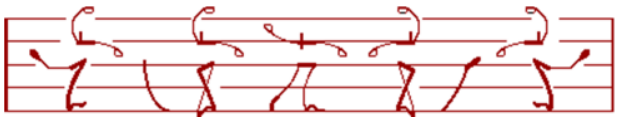

Fig. 3 An Example of Sutton Movement Writing [31]

Dance notation rapidly became more elaborate, more detailed, and more formal, leading to highly detailed notations that needed lengthy training to read and write fluently, such as 'labanotation' [27] (Figure 2).

In the late twentieth century, however, we see a return to a form of pictorial notation for dance and other forms of movement, the Sutton Method [28]. Waters and Gibbons assert that the Sutton method conveys as much information as the Laban notation, but that dancers find it much easier. What have they to say about this progression? They have an interesting and insightful section on the interaction between 'design language' and 'notation system', and they correctly observe that a good notation leads to fruitful innovation, but they have nothing to say about the progress and lifecycle except that as the design language becomes refined and extended, it leads to developments in the notation, which lead to further developments in the design language, and so on.

Algebraic Notation. Stallings [29] describes three major stages in the development of our schoolroom algebra, starting with 'rhetorical algebra' in which operations were described in words:

> If the instance be, "ten and thing to be multiplied by thing less ten," then this is the same as if it were said thing and ten by thing less ten. You say, therefore, thing multiplied by thing is a square positive; and ten by thing is ten things positive; and minus ten by thing is ten things negative. You now remove the positive by the negative, then there only remains a square. Minus ten multiplied by ten is a hundred, to be subtracted from the square. This, therefore, altogether, is a square less a hundred. . . [Nelson, 1993, p 33]

Or, in modern terms, $(x + 10)(x - 10) = x2 - 100$. The steps to our modern 'symbolic' algerba were gradual. The equality sign was developed by Robert Recorde in 1557, who chose two parallel lines of the same length because "no 2 things can be more equal" – a *picture* of equality. Much earlier, Stallings writes, the Rhind papyrus of c. 1650 BC contains "one of the earliest known symbols for addition and subtraction; a pair of legs walking forward, which denoted addition and a pair of legs walking away for subtraction (Eves, 1983)." Once again, we note that these early symbols are *pictures*.

Staff Notation of Music. Rastall (1997) presents an excellent overview of the development of Western musical notation. The common notation for Western music has undergone a very long course of development and the notations of its earliest stages are not well understood; but in the 9th and 10th centuries, we find the earliest instances of the use of 'neumes' as musical signs, in the manuscripts of the Swiss monastery of St Gall. "The earliest neumes were inflective marks which indicated the general shape but not necessarily the

exact notes or rhythms to be sung" (wikipaedia, under *neume*). Their function was to remind their reader of the essential features of a melody that has already been learnt, rather than to specify the performance exactly. Although neumes were already highly developed by the time of the St Gall manuscripts, their appearance is at least partly a picture of the trajectory of pitches: the neume called 'climacus', for instance, stands for three notes descending, and looks exactly like that:

Fig. 44 The climacus neume – three notes descending

The development of neumes is one of increasing sophistication, from simple lines and dots to fairly elaborate flourishes. By a very complex series of developments, notation eventually arrived at our familiar staff notation, far more precise and detailed than the neume – though even here, we can distinguish different levels of detail, depending on the genre. Today, three notes descending might be written like this:

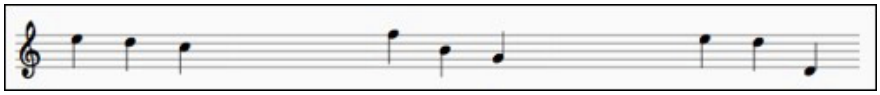

Fig. 55 Three different realisations of "three notes descending"

Unlike neumes, present-day notation offers many possible elaborations, which can give much more precision about how the notes are to be played:

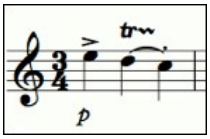

Fig. 66 Three descending notes with more precise information about technique

Fig. 77 one of Wolff's quasi-pictorialsymbols

Rastall distinguishes eight kinds of notation in music, from a pure aide-mémoire, through skeleton notations to more and more closely detailed notations and then to 'inspirational' notations where "visual symbols or ideas expressed graphically and/or in words inspire the performer to certain actions". He no doubt had in mind such notations as the quasi-pictorial symbols bused by Christian Wolff [32] (Figure 7).

Wolff's symbols mean things like 'make a sound in a middle place, in some respect, of the sounds around it,' make a sound involving stretched material', or 'play after a previous sound has begun, hold till it stops'. We see here a move away from the high level of formality and precision that had been accumulating in music notation and a return to the under-specified world of the neume.

What have we learnt? Three very different fields of notation have been presented, and in each case we have seen that the earliest forms were pictorial or picture-like; that the notations developed in degree of formality and in degree of expressiveness and sophistication; and eventually, in at least the dance and music fields, a new notation was presented, in which the level of formality was abruptly reduced.

We have also seen that in all three fields there is no proper account of the development. Nothing is said about the fact that the earliest stage was usually pictorial, and nothing is said about the development except that it needed to become more expressive.

Above all, there is *no awareness that the course of development in each field is profoundly similar*. The evidence is that notations start and develop in similar ways, whatever field they may be designed for. The purpose of this paper is to make a start at describing the common features of notational development.

1.2 What is a 'good' notation?

Although the field of human-computer interaction offers many examples of detailed analysis, there are very few attempts to provide evaluation techniques suitable for notations as a class. One such is the 'cognitive dimensions' framework; another is the so-called 'physics of notations' [33].

Cognitive Dimensions of Notations. The cognitive dimensions framework [2] sets out to provide a useful vocabulary with which to describe important aspects of notational systems at the structural level – that is, it explicitly does not deal with details of rendering and presentation. It is important to note that because this framework is much concerned with creating or making changes to documents, what is considered is the notation in the environment of its editing system, which might be an IDE or might be pen and paper or might even be marble and chisel. Examples of dimensions are *viscosity*, meaning that local changes need many actions to accomplish: changing all the first-level headings of a document to use larger font, for instance, can be a tedious job unless a style-sheet system is available. A style-sheet, however, introduces the dimension of *abstraction level*: some users find abstractions to be a serious barrier. A third dimension is *hidden dependencies*, where it is difficult to know what will happen if one object is changed because it is not manifest whether other objects will be affected; changing one cell in a spreadsheet, for example, may have big knock-on effects. About 16 dimensions are included in the framework.

Within the framework, no dimension is considered to be evaluative on its own, but only in the context of one of the half-dozen archetypal 'activities': *incrementation* (adding more objects to a structure, eg defining a new class), *modification* (changing the structure, eg rebuilding the class hierarchy), *transcription* (e.g. turning arithmetic procedures into program code), *search*, *exploratory design* (working out how to write a program, composing a piece of music, etc), and *exploratory understanding* (discovering the internal structure and workings, e.g. of a piece of software). If we know the typical activity that a notation will be used for, we can then make some evaluations: a notational system with high viscosity is more suitable for transcription activities than for exploratory design, for instance.

The framework does not claim to offer precise or even vague quantitative predictions, but instead to offer 'tools for discussion' which can lead to improvements in existing notational systems. It has been applied with success in many fields, but unfortunately it offers no clue as to the lifecycles of notations. It is a purely synchronic evaluation.

Physics of Notations. Moody [33] claims to have developed 'principles for designing cognitively effective visual notations', optimised for human communication and problem-solving. These principles are such as semiotic clarity, meaning that there should be a 1:1 correspondence between semantic constructs and graphical symbols. While it is unclear to us that the principles proposed by Moody merit a comparison to physics, what is clear is that this is another purely synchronic form of evaluation, with nothing to say about how notations might develop.

2 The Notational Life Cycle

We propose that notations generally proceed along a spiral path in five stages, as in Fig. 8:

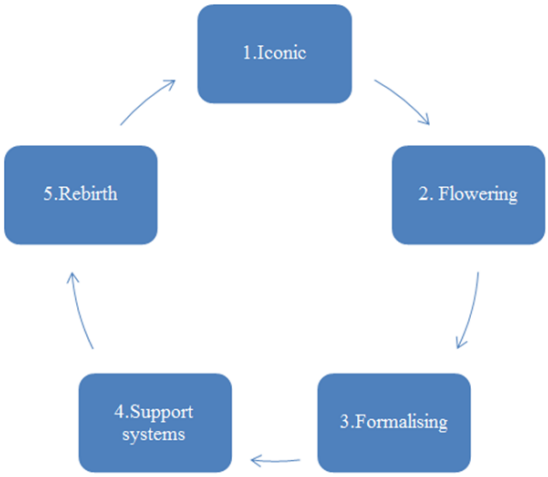

Fig. 8 The notational lifecycle

Each notational stage will have different characteristic problems. (These 'stages' are a convenient idealisation, of course, just as the usual colour names – red, orange, yellow ... – are a convenient idealisation of continuously changing wavelengths and mixtures of wavelengths.) Every time we see a new cycle start, by creating a new notation to replace an older one, we see the stage 5 problems exchanged for the stage 1 problems. Stage 5 problems include an over-stretched vocabulary with many special cases; on restarting at stage 1 the vocabulary will be 'higher-level' [2] with new rules for combining components more productively, but the environment will be worse and little attention will be paid to issues of any type of activity except 'incrementation' [2], i.e. adding material. The notational stages are:

Iconic. When a new notation is created by an unsophisticated person, it will be pictorial or iconic, like early circuit diagrams (Figure 9).

The creator imagines how to add items (more wires or resistors) but does not imagine how to make large changes; likewise personal use is imagined, but not collaborative use. In terms of the 'cognitive dimensions' framework [2], [3], high viscosity and frequent hidden dependencies are acceptable for personal use. Outside that framework, there are other types of consequence: the notation will probably be very incomplete, because the notation is more of an *aide-mémoire* than a complete description.

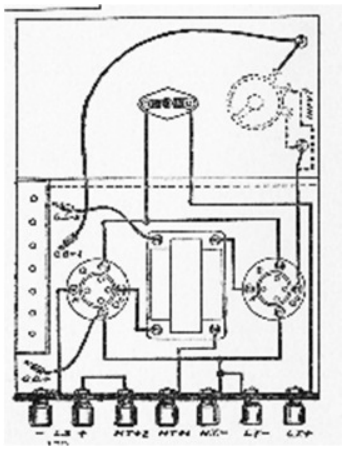

Fig. 9 Early circuit diagrams [34]

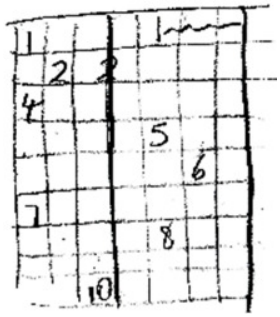

Fig. 10 [Private communication to first author.]

Figure 10 shows a notation devised by a novice to show how to ring a tune on handbells. The bells are

numbered 1-10 and they are struck (by different people) in the order specified from left to right. The heavy central line is a bar line (US: measure line). The figure 2 across the line is an attempt to indicate a dotted note (one-and-a-half beats), and the squiggle at the top right shows a prolonged shake of the bell. The reader is expected to know which bell sounds what note. This notation is at the start of stage 1; already it is being stressed by the need to accommodate additional features, such as dotted notes. Notice that it allows a deal of slop: bell 3 has been accidentally overlooked, and the second bar (measure) has 4 beats compared to the 3 beats of the first bar.

Flowering. The notation becomes popular. It is used by other people for slightly different tasks. The vocabulary has to be increased, and consistency falls. As time passes, the early instances of its use need to be updated, and maintenance is slowed by the hidden dependencies and viscosity. Collaborative use increases and forces users to devise techniques for secondary notation to improve the role-expressiveness; these techniques may arise differently and incompatibly among different user groups. The notation is a victim of its success, and is stressed by unforeseen problems.

Formalising. To improve consistency, rules are made explicit. The vocabulary is defined, the syntax is defined, and the possible relationships between entities are defined. The notation gets harder to use, although it's more likely that a syntactically correct document means what its creator intended. Labanotation is at this formalised stage, for example.

Support Patchwork. A patchwork of tools appears to support users. For example, because the notation has become more formal, it has become harder to sketch casually, so another notation is introduced as a sketching tool, before transcribing the sketch into the target notation, which we call 'decoupling'. A typical example is the development of graphical notations for sketching designs for software before starting implementation – such as UML, or at a simpler level the humble flowchart. These are what Waters and Gibbons (see above) called 'design languages', in that they free the user from some of the constraints and details. Typically they also free the user from order constraints, because ideas come to the mind in an order dictated by their internal logic rather than by the demands of a formal notation. That is to say, if a diagram is in use as a sketching tool, new logic can be added to it at any point, arbitrarily, whereas if a class hierarchy is being created in an IDE it is usually impossible to create subclasses until their superordinate class has been created: thus, the diagram allows free working, whereas the IDE forces the expression of the program to take place in top-down order.

Rebirth. The vocabulary gets too large and is obviously too low-level. The uses it is put to become more complex, and the structures built in it grow so fat that viscosity, hidden dependencies, and poor visibility are a serious impediment. The crisis is resolved by creating a new, higher-level notation. This will have its own problems, so we are back at stage 1 but at a higher level.

The Sutton method, a replacement for labanotation, is an example of rebirth. Similar examples can be found in the development of various software systems, such as programming languages, CSS, and content-management systems; also in other domains, such as circuit diagrams and music notation.

The proposed stages represent the straight track. However, there are lots of factors that might affect the straight track. We proposed 5 factors that may affect the straight track:

Repurposing. Notations are augmented for a specific purpose. For example, CSS originally described the spatial characteristics of typeface, which might include some positional aspects; it is now the primary tool for laying out the spatial structure of a webpage.

Change in Demand. Due to cultural or environmental influences, users may be more sophisticated; new groups of users may appear. What they want may change, forcing adaptations.

Changes in World. Inventions could change the role of the usage of some notations. Thus, the notation has had to change dramatically, and many different notations have been created by notational novices, frequently starting at stage 1. For example, after introducing the predicate logic, many notational systems were developed.

Mature Disfluency. Certain notations are so designed that their users' environments slowly become damaged, and the more the notation is used, the worse the damage. An example is the accumulated silt in everyone's computer: dot-files, preference-files and user-manuals for applications long since discarded continue to consume space and efficiency.

Market Forces. Notations do not exist in isolation, and sometimes two notations have the same domain. One possible outcome is simple rivalry, as with Leibniz's and Newton's notations for calculus. Another possible outcome is fusion, as in the Unified Modelling Language. The third possible outcome is sophisticated, forward-looking rivalry, in which the creators of one notation attempt to pre-empt rivals by capturing as much notational space as possible.

3 A Pedagogical Example: Constraint Diagrams and their Life Cycle

To understand our proposed notation lifecycle, we provide a detailed example of the development of an existing notation. The Constraint Diagram notation was chosen because many developments happened in a short time.

At stage 1, the notation was designed for one person doing one small job and thus the anticipated activities on constraint diagrams were limited to 'incrementation' – adding more of the same objects. The notator did not look forward to anticipate modifications or other activities, and worked at an individual level, rather than collective. This stage produced Fig.11.

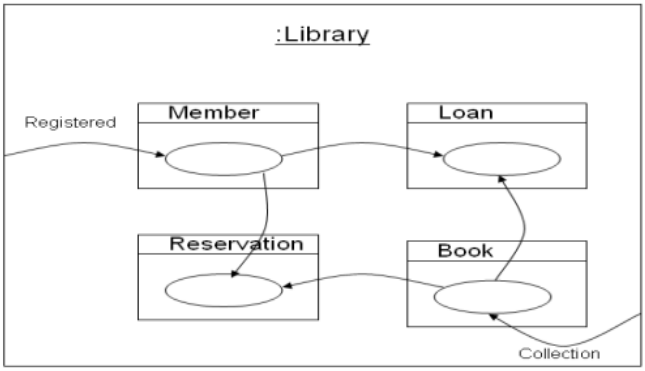

Fig. 11~~1~~. Stage 1 of Constraint Diagram [4]

The notation became popular and was used by other people for slightly different tasks such as behavioural specification [5], [6]. Thus, the vocabulary increased, leading to stage 2 of constraint diagrams as shown in Fig.12. Here we see the notation flowering.

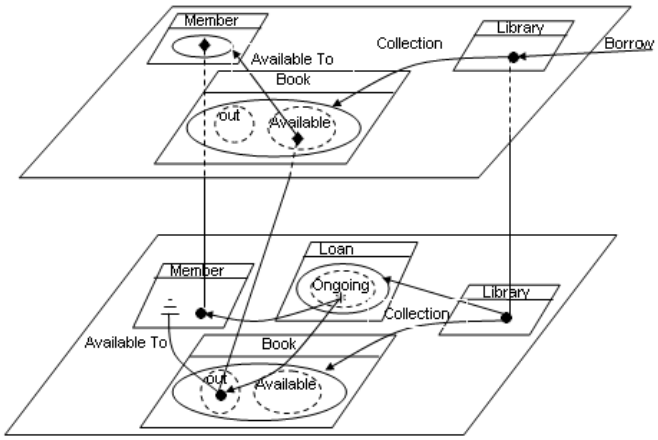

Fig. 12~~12~~. Stage 2 of Constraint Diagram [5]

In Stage 3 as shown in Fig. 13, an attempt was made to improve consistency by making rules explicit [7]. The vocabulary and the syntax were defined [8]. The notation started to get harder to use and users would be categorised as experts or novices; a classic example of the formalised stage of a notation cycle.

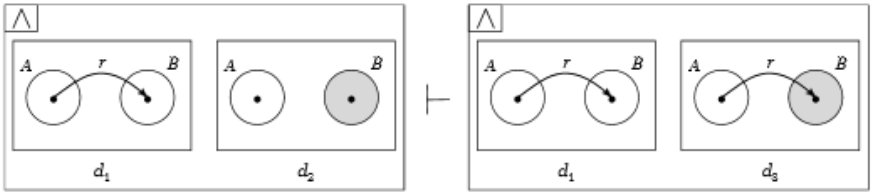

Fig. 13~~13~~. Stage 3 of Constraint Diagram [7]

At stage 4 as in Fig. 14, the constraint diagram notation was rigid and thus the constraint tree notation [11] was introduced as a complementary notation. Here we see the 'support systems' stage of the lifecycle.

This attempt was improved by introducing reading trees; both explicitly [12], [13], [14], [15], [16] and implicitly [17]. There was another attempt [18] of improving this notation using the implicit reading tree with the influence of Z notation [19].

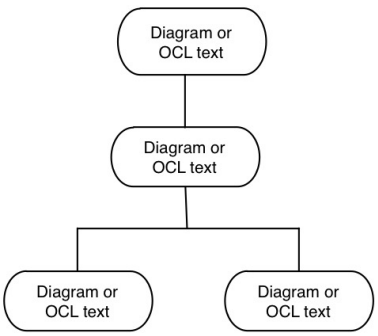

Fig. 14~~14~~. Stage 4 of Constraint Diagram [11]

At this stage, the vocabulary got too large and was obviously too low-level. Thus, new notations needed be devised at a higher level of idealization such as Concept Diagrams [20], [21], [22] as in Fig. 15. This is a start

of a second life cycle, the 'rebirth', in which we return to stage 1 at a higher level.

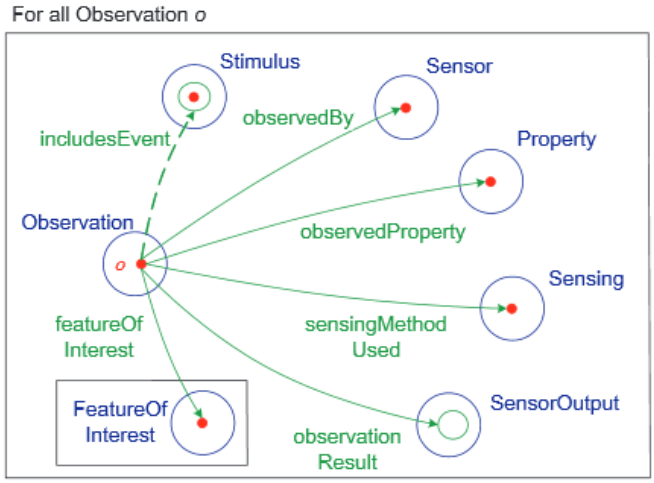

Fig. 15~~15~~. Stage 5 of Constraint Diagram [22]

4 Conclusion

In this paper we have argued that notations in very disparate fields develop in very similar ways, and that their paths can be characterised by our five-stage model. To our knowledge, this is an entirely novel observation. The stages have been closely illustrated in the account of the development of constraint diagrams.

Does this lead to useful advice for the designer of a notation? Both the cognitive dimensions framework and the 'physics of notations' approach (see above) can offer some useful advice: for instance, the former can advise a notation designer to think in terms of activities beyond incrementation, and to beware of, say, hidden dependencies in some activities (modification, exploratory understanding) though not all activities (transcription), while the 'physics' approach can remind the designer that symbols should not be overloaded. Yet such advice is purely synchronic.

It is too early know what benefits, if any, may come from viewing notational design diachronically rather than synchronically. However, in this paper, we have focused on their evolution, trying to postulate the factors that cause a notation to develop through different stages. This leads to some principles for notational design that we believe are novel, in that they are based on the likely evolutionary path of a notation:

1. Don't try to create the perfect notation for all time. However perfect the notation might be for the present moment, things will change.
2. Therefore, do try to design a notation that can easily be extended to new activities when they become necessary.
3. Prepare for the patchwork of support that will become necessary, by providing whatever hooks may be useful – but in the early stages it may be difficult to foresee the need.
4. Try to short-circuit the evolution by starting at level 2, the flowering stage, rather than level 1, the iconic stage.

It will be obvious that these recommendations are at present quite untested, and we hope that future research on the histories of notations will create a more detailed picture and will correct our thinking where necessary.

References

1. Crystal, D.: The Cambridge Encyclopedia of Language. Cambridge University Press (1997)
2. Green, T. R. G.: Cognitive dimensions of notations. In R. Winder and A. Sutcliffe (Eds), People and Computers V. Cambridge University Press (1989)
3. Green, T. R. G., Petre, M.: Usability analysis of visual programming environments: a cognitive dimensions

framework. J. Visual Languages and Visual Computing vol. 7, pp. 131-174. (1996)
4. Kent, S.: Constraint Diagrams: Visualizing Assertions in Object-Oriented Models. Technical Report ITCM97/C2, University of Brighton (1997)
5. Gil, J., Kent, S.: Three dimensional software modelling. Proceedings of the 1998 International Conference on Software Engineering. (1998)
6. Gil, Y., Howse, J., Kent, S. (1999) Constraint Diagrams: a step beyond UML. In: Proceedings of Technology of Object-Oriented Languages and Systems (TOOLS 30), IEEE Computer Society Press, Los Alamitos, pp 454-463
7. Burton, J., Stapleton, G., Hamie, A.: Transforming constraint diagrams. In: P. Cox, A. Fish, J. Howse (Eds.), Visual Languages and Logic, pp. 62-80. (2009)
8. Gil, J., Howse, J., Kent, S.: Towards a Formalization of Constraint Diagrams. In: Proc. of IEEE Symposia on Human-Centric Computing (HCC '01), pp. 72–79, IEEE press. Stresa, Italy. (2001)
9. Gil, J., Howse, J., Kent, S.: Formalizing Spider Diagrams. In: Proceedings of 1999 IEEE Symposium on Visual Languages (VL99), IEEE Computer Society Press, Los Alamitos, pp 130-137. (1999)
10. Howse, J., Molina, F., Taylor, J.: On the Completeness and Expressiveness of Spider Diagram Systems. In: Proceedings of Diagrams 2000, lecture Notes in Artificial Intelligence, Springer. (2000)
11. Kent, S., Howse, J.: Constraint Trees. In Clark, T., Warmer, J. (Eds), Object Modeling with the OCL, Springer Berlin-Heidelberg, Lecture Notes in Computer Science, vol. 2263, pp. 228-249 (2002).
12. Fish, A., Howse, J.: Computing Reading Trees for Constraint Diagrams. In Proceedings of Applications of Graph Transformations with Industrial Relevance conference, Virginia, USA, Lecture Notes in Computer Science 3062, pages 260-274, Springer. (2003)
13. Fish, A., Flower, J., Howse, J.: A reading algorithm for Constraint Diagrams. In Proceedings of IEEE Symposium on Human-Centric Computing, Languages and Environments, Auckland, New Zealand, IEEE, pages 161-168. (2003)
14. Fish, A., Masthoff, J.: Do monkeys like elephants or do elephants watch monkeys? An empirical study into the default reading of constraint diagrams. Technical Report VMG.05.2, University of Brighton. (2005)
15. Fish, A., Flower, J., Howse, J.: A Reading Algorithm for Constraint Diagrams. In: Proceedings of HCC03 (2003), 161-168.
16. Fish, A., Howse, J.: Computing Reading Trees for Constraint Diagrams. In: Proceedings of AGTIVE03 (2003)
17. Fish, A., Howse, J.: Towards a default reading for Constraint Diagrams. In Proceedings of the 3rd International Conference on the Theory and Applications of Diagrams 2004, Cambridge, UK, Lecture Notes in Artificial Intelligence 2980, pages 51-65, Springer. (2004)
18. Howse, J.,Schuman, S,: Precise visual modelling. SoSyM 4: 310-325. 2005)
19. Woodcock, J., Davies,J.: Using Z: Specification, Refinement, and Proof Prentice-Hall.(1996)
20. Chapman, P., Stapleton, G., Howse, J., Oliver, I.: Deriving sound inference rules for concept diagrams. In: Proceedings of the IEEE symposium on visual languages and human-centric computing 2011, Pittsburgh, PA, USA. (2011)
21. Stapleton, G., Howse, J., Chapman, P., Oliver, I., Delaney, A.: What Can Concept Diagrams Say?. The 7th International Conference of Diagrammatic Representation and Inference. In Lecture Notes in Artificial Intelligence no. 7352, Springer, pages 291-293.(2012)
22. Howse, J., Stapleton, G., Taylor, K., Chapman, P.: Visualizing Ontologies: A Case Study. International Semantic Web Conference. In Lecture Notes in Computer Science no. 7031, Springer, pages 257-272. (2011)
23. Fetais, N., Cheng, P. C. -H.: Analysing Existing Set Theoretic and Program Specification Diagrams. Qatar Foundation Annual Research Forum Proceedings, vol. 2012, Doha, Qatar. (2012)
24. Design Languages, Notation Systems, and Instructional Technology: A Case Study Author(s): Sandie H. Waters and Andrew S. Gibbons. Source: Educational Technology Research and Development, Vol. 52, No. 2.(2004)
25. Arbeau, T. (1589) Orchésographie. Translation by M. S. Evans (2012), The Noverre Press.
26. Labanotation: http://dancenotation.org/ (accessed 20 July 2015)
27. Laban, R.: Laban's Principles of Dance and Movement Notation. 2nd edition edited and annotated by R. Lange. London: MacDonald and Evans. (1975).
28. Sutton, V. (1997) Dance Writing Shorthand for Classical Ballet. Center for Sutton Movement Writing (http://www.movementwriting.org/)
29. A Brief History of Algebraic Notation, Lynn Stallings School Science and Mathematics, Volume 100, Issue 5, pages 230–235. (2000)
30. Rastall, R.: The Notation of Western Music: an introduction. Leeds: Leeds University Press. (1997)
31. Sutton, V., http://www.valeriesutton.org/. (accessed 20 July 2015)
32. Wolff, C.: For 1, 2 or 3 people. Edition Peters. (1964)
33. Moody, D. L.: The Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Trans. Software Eng,. 35, no. 6, 756-779. (2009)
34. Camm, F. J. : Newnes Wireless Constructor's Encyclopædia. London: Newnes, n.d. p. 179