Galactic Dimensions: A Unifying Workstyle Model for User-Centered Design

Pedro Campos and Nuno J. Nunes

University of Madeira,
Campus da Penteada, 9000-390 Funchal, Portugal
{pcampos,njn}@uma.pt

Abstract. This paper describes a new unifying workstyle model for the user-centered design process, comprised of eight dimensions that we claim as fundamental to supporting the UCD process. Our proposal is new because it is the first workstyle model tailored to UCD. We also show the usefulness of workstyle modeling when evaluating the stage/effort of a project at a given time. Our workstyle model was based on the identification of the main obstacles to UCD and SE integration, current research results and extensive observation of HCI students involved in UCD projects. Though simple, it models the designer's behavior and can be effectively and easily used to (a) choose adequate tool support for a given phase of a project and (b) drive the development of new UCD tools.

1 Introduction

User-Centered Design (UCD) is a process that fosters the participation of users in designing and evaluating a system, in order to obtain products that are better suited to users' expectations. However, after almost two decades of UCD tools and techniques research, its adoption remains limited to large organizations and practitioners who recognize its value [6].

Despite all the research efforts dedicated to bringing better tools to the industry, designers still consider tools don't meet their needs [8]. After more than a decade, CASE tools have not been widely used [7], although the market is rapidly growing [6]. Despite of limited CASE tool adoption, there is evidence that the technology improves, to a reasonable degree, the quality of documentation, the consistency of developed products, standardization, adaptability, and overall system quality [4]. It has been argued that future tools should be based on sound models of a software process and user behavior, and should support both creative aspects as well as rigorous modeling [6].

Workstyle modeling [14] has been proposed as a technique to record the interaction style of a group of collaborators during any software development activity. UCD is an iterative, evolutionary process. This means designers often engage in different workstyles as they iterate towards the final design. This is why we claim that modeling the styles of work can be particularly useful in UCD. It has been widely recognized that current User Interface (UI) tools don't support the designers' activities, in particular it has been argued that UI tools suffer from limited combinations of threshold ("how hard is it to learn?") and ceiling ("how much can be done?") [10]. There is clearly room for benefits here, since it has been shown that a UI development tool has the potential to influence between 50% and 70% of the application code [10].

Constantine and Lockwood [2] described their ideas of "galactic dimensions" as a metaphor change towards fully interconnected and synchronized visual development tools. Traditional CASE tools were based on a metaphor referred to as the "glass drawing board", since they merely represented two-dimensional paper models on the glass surface of a monitor. The "glass galaxy" was then proposed as a multidimensional problem-solving space in which developers could drill down into objects in one dimension, and be taken via software "worm holes" to another. Clicking on a use case could take the developer to its definition in a glossary. Selecting that use case could also show the abstract components that support it, or the concrete widgets for a given realization of that model. Even entries in help files could be linked to the user roles they support, or to the actual code and visual UI controls.

We take this idea further and argue for CASE tools supporting "galactic" dimensions: tools that not only support fast accelerated development through traceability and integration, but also are able to rapidly adjust to any given workstyle in a transparent way. We propose a new workstyle model comprised of eight "galactic" dimensions that we consider as fundamental to supporting the UCD process. Our proposal is new because it is the first time workstyle modeling is applied to UCD and our model can be used to estimate the stage/effort of development in a graphically intuitive way. Another contribution of our paper is that we also show how effective workstyle modeling can be to drive the development of new UCD tools or to choose tool support for a given project phase.

Our workstyle model was based on the identification of the main obstacles to UCD and SE integration [12], current research results and extensive observation of Human-Computer Interaction (HCI) students involved in a UCD project. It incorporates two renamed dimensions from [14], (Asynchrony and Distribution), and two of the modeling-style dimensions from [13] (Perspective and Formality). We introduced three more dimensions (Detail, Functionality and Traceability) and unified these dimensions into a common, coherent model, where the axes fall into one of these categories: collaboration style, notation style or tool-usage style.

This paper is organized as follows: in the next section, we describe related work on models and taxonomies for software design and UI design. Section 3 describes our new workstyle model for UCD and Section 4 illustrates how it graphically conveys information regarding the stage of development of the UCD process. In Section 5, we apply our model to several representative tools and claim that current successful tools are those that can better adjust and support any region of our model. Finally, in Section 6, we draw some conclusions and outline work that might bring benefits to both SE and UCD.

2 Related Work

Software design (which includes interaction design) is often a team activity and most projects involve stakeholders with different backgrounds that must cooperate in many different and interrelated activities.

Wu and Graham [14] describe a novel model for recording the working style of people using an interactive system. Workstyle modeling complements task modeling by providing information on how people communicate and coordinate their activities, and by showing what style of artifact is produced.

The workstyle model was developed in the context of the Software Design Board project, a project aiming to provide better tools for software design. The model was validated through evaluation of existing design tools, and motivated the design of a new software design tool. It is comprised of eight axes: four of them describe collaboration style (Location, Synchronicity, Group Size and Coordination); the remaining four describe the nature of the artifact being produced (Syntactic Correctness, Semantic Correctness, Archivability and Modifiability).

The workstyle model for software design has the advantage of being simple to apply and clearly showing where a tool can fail to match the intended work context. However, it is not sufficient for capturing UI specific activities. Transitions (or shifts) in the workstyles of interaction designers are more frequent and more intense than in any other software design activity.

Traeetteberg [13] claims that current UCD tools should be able to cover a three-dimensional cube of Perspective (moving from problem/requirements space to the solution/design), Granularity (from high level to low level) and Formality (formal, i.e. machine-understandable versus informal i.e. context-dependent). It is suggested that in the course of designing an interface, several languages have to be used to cover all the needed perspectives (see Figure 1).

As an example, Traeetteberg [13] suggests one interpretation of the granularity level across different perspectives, as can be seen on the left side of Figure 1. A task is performed to achieve a goal, and is often supported by a specific component composed of dialogue elements, which is placed in a pane containing widgets.

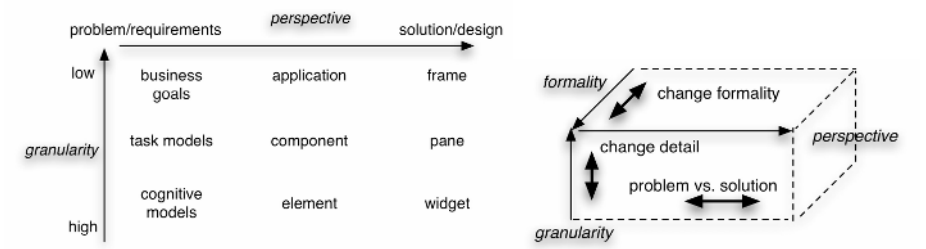

Fig. 1. Left: an interpretation of the granularity versus perspective. Right: movements in the representation framework of Traeetteberg [13].

Cognitive Dimensions [5] have been proposed both as an evaluation technique for visual programming environments and as a discussion tool for designers. This technique concentrates on the activities rather than the finished product. However, it is limited to the information artifacts of visual programming languages.

Prototyping techniques still leave a considerable gap between the inception level models of user intentions (task cases, use cases, scenarios and other requirements level models) and the concrete user interface. The center ellipse in Figure 2 illustrates this gap. A growing awareness of this conceptual gap lead Constantine and colleagues to develop a new language for visual and interaction design, called Canonical Abstract Prototypes [2]. This language fills the gap between existing inception level techniques, such as the illustrated UML-based interaction spaces or visual content inventories, and construction level techniques such as concrete prototypes.

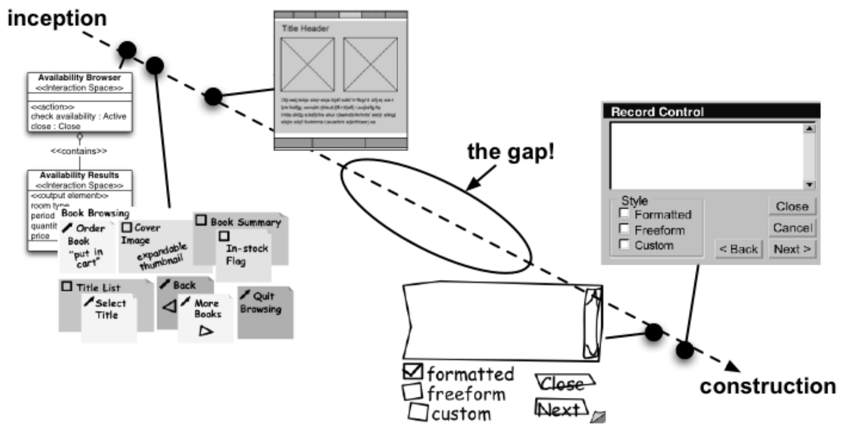

Fig. 2. Prototyping techniques from inception to construction (adapted from [2])

All of these models support the need for a unifying reference framework under the context of UCD processes. They also reflect the importance of considering these several dimensions into the design process of UCD tools. However, these models are too specific and are not expressive enough to be applied more generally to UCD. This is why we expand them and combine the most significant dimensions of some of them: to obtain a more useful and usable model that has more meaning by conveying more information in a better way. In the next section, we will describe our new workstyle model and illustrate its applications.

3 A Workstyle Model for User-Centered Design

There are eight continuous axes in our “galactic” workstyle model for UCD. These axes are grouped under three main categories:

- **Notation** style-related dimensions (Perspective, Formality and Detail),
- **Tool usage** style-related dimensions (Traceability, Functionality and Stability) and
- **Collaboration** style-related dimensions (Asynchrony and Distribution), as shown in Figure 3.

In this section, we briefly describe each of these dimensions and provide a set of questions that can act as guidelines to apply the model to tools, notations or, in general, styles of work adopted by interaction designers.

Perspective. This axis plots the perspective, or view, of the artifact being developed. **Questions:** is the notation capable of expressing business goals? Or non-functional requirements such as customer experience requirements? Does it help define the purpose of the system? Does it describe interaction aspects of the system? How close is it to the final product?

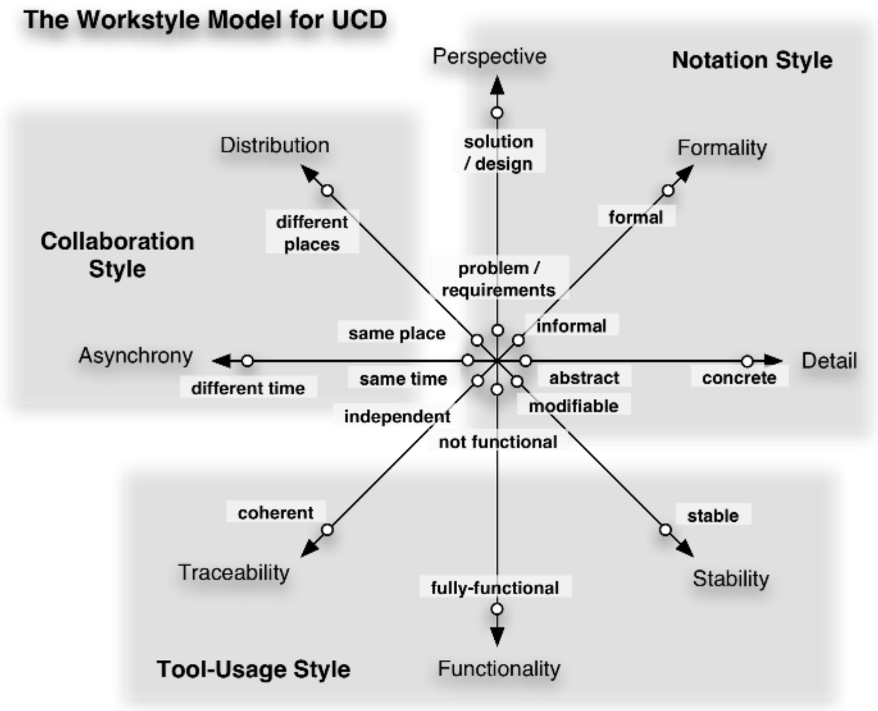

Fig. 3. A unifying workstyle model for user-centered design

Formality. This axis classifies the workstyle of a designer creating artifacts in a formal vs. informal way. In the early stage of the process, designers use rough, ambiguous sketches to freely express ideas quickly [8]. This workstyle also fosters comparison of design alternatives and creativity, since the uncertainty of sketches encourages the exploration of design ideas. As design progresses, a more formal style of work is incrementally adopted, as designers need to focus on the precise meaning of their models. An example of this shift is moving from a whiteboard to a CASE tool. **Questions:** how easy is it to define rough ideas? Does the meaning matter? Does the notation force you to use a rigid syntax/semantics?

Detail. We added this axis to plot the level of detail (or abstraction) the designer is working at. High-level, abstract models facilitate problem solving in organization, navigation and overall structure of the UI, leaving aside the details. On the other hand, realistic (or figurative) prototypes address high-detail design issues [3]. Disciplined designers tend to assume a workstyle that goes from higher-level abstract representations towards more realistic and detailed representations as the process evolves [3].

Questions: can you abstract irrelevant details using the notation? Can you think about navigation and structure of the overall interaction using the notation? Can you incrementally add enough detail?

Stability. This dimension describes how difficult/frequent it is to modify any aspect of the artifact(s) being developed. A content inventory of the UI modeled in a UML tool is highly modifiable because it is easy to change names, positioning, size and other aspects of the elements. This is opposed to drawing a model of the UI with pen and paper, since changes are harder to accomplish. Brainstorming, for instance, is a very unstable workstyle because changes are very frequent. High values in this axis indicate less frequent or less significant changes.

Questions: How easy is it to modify previously created artifacts using the tool? How frequently do you make those changes? Are there particular changes difficult to accomplish with the tool?

Traceability. This is a new dimension we introduce. It describes if the elements of the artifact being developed are consistent and interconnected (thus being highly traceable) or if they are completely unrelated and independent. As an example, developers might adopt a workstyle in which they choose to keep links from task cases steps and the concrete UI widgets that implement those task steps. In this case, it is possible to trace a task step to the concrete widget and to trace a widget to the task step it implements. This dimension is closely related to stability and the number of artifacts produced during a project. As they increase, traceability becomes more important.

Questions: Are you using the tool to maintain interconnections between model elements? How important is it to navigate through your model? Does the tool maintain several different views in a synchronized way (e.g. design view and code view)?

Functionality. This is also a new dimension we introduced. It represents how much functionality is being addressed (by using the tool to build a prototype). There is a barrier between software engineers and usability professionals regarding this matter: software engineers are engaged into building reliable, functional systems, leaving user-friendliness to the usability specialists. Usability and interaction designers, on the other hand, first design and test the interface with end-users, leaving implementation to software engineers, regarded as functionality builders. Those two processes should not be separated [12] and considering this dimension will help overcome that barrier. This dimension is also important because designers combine visual design (presentation issues) with interaction design (behavior issues).

Questions: How much functionality, behavior and dynamics can you add to your prototypes using the tool? How easy is it to test the interaction by using the tool?

Asynchrony. This axis refers to the collaboration style that designers assume: they can make changes to the work being developed at the same time (a *synchronous* workstyle) or they can work at different times (engaging in an *asynchronous* workstyle) [14]. The higher the value in this axis, the more asynchronous is the workstyle.

Questions: do the team members change artifacts at the same time? Or do they make changes at different times? How frequently?

Distribution. This dimension describes whether work is being conducted at the same physical location or at geographically distant locations.

Questions: how far are the team members collaborating? Are they in the same building? Or are they in a different continent, or scattered through a country?

These dimensions can be effectively used to assess a given workstyle adopted by an interaction designer or a team. A single workstyle is plotted as a line (a point in the eight-dimensional space) whereas regions (or planes) represent sets of workstyles.

4 Using the Model to Assess a UCD Project Effort

In general terms, it seems reasonable to say that as the process evolves, designers tend to assume a workstyle that spreads them away from the center of our model.

Under the *perspective* axis, for example, it is clear that as time goes by, developers move from the domain/problem level towards the solution/design space. In terms of *formality*, they start out with informal, ambiguous sketches and move to formal languages later on, when coding increases and functionality becomes more important. Under the *detail* dimension, as we have already seen, skilled designers tend to go from higher-level abstract representations towards more realistic and detailed representations as the process evolves.

Also, as the deadline approaches, prototype *functionality* is added (and is needed for user testing and customer delivery). In an initial phase, designers don't spend much effort on functionality: it is more important to rapidly compare design alternatives. In addition, as time goes by, ideas start to solidify and changes become more incremental, rather than dramatic (thus increasing *stability*). Under the *traceability* axis, the number of artifacts increases, and so does the number of inter-connections between them, which motivates the need for increased traceability. This also happens because developers are not interested in throwing away the models (as in brainstorming) but rather in keeping all models created so far in a coherent state.

Under a collaboration style perspective, the transition may not be so straightforward. Nevertheless, as development tasks get larger and work allocation is made, there is a tendency to work asynchronously and at different places (thus increasing the workstyle value on the Asynchrony and Distribution dimensions), because developers feel the need to focus and split work.

As we will see in Figure 4, one of the advantages of our model is the fact that it graphically conveys implicit information regarding the temporal stage of development (again, in general terms). Figure 4 shows the workstyle model plotted for several phases of a UCD process (in this case the Wisdom process [11]). This is a rough modeling of the styles of work and how they vary according to the activity being performed. In the inception phase, all workstyle dimension values are low: developers think informally in terms of requirements, at the same time and place, without functionality issues in mind, doing many changes to compare alternatives. As they move to elaboration and construction, they start to adopt a workstyle with higher values in all dimensions (although some more than other).

The grey area shows the effort along the time for each activity. The circles show the workstyle adopted by developers along the time as well. When the effort is higher, workstyle transitions become more frequent (the circles thickness plot the frequency of transitions).

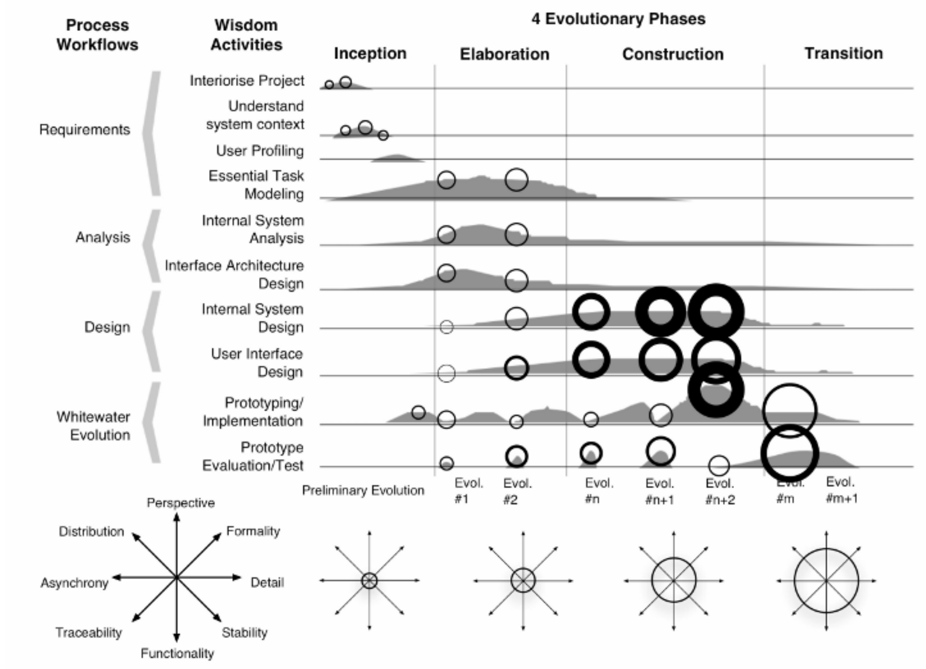

Fig. 4. The workstyle model plotted along the different phases of a UCD process

In practice, this evolution is never translated into a perfect circle, as different projects have different goals and needs. For instance, the larger the organization, the greater the formality and location dispersion. However, if we think in terms of workstyle iterations, as design evolves, there is a general tendency to move away from the center of our model. Also, if regions (instead of lines) become plotted in our model, we have an indication that iterations and workstyle transitions become more frequent, which accounts for higher project effort. Consequently, by checking the plotted regions' size, one can estimate how much effort is being put by a team of UCD developers. The power of our model (over other models such as [5, 13, 14]) is the fact that it can be used to assess the level of development in a graphically intuitive way. An image can convey more information than words or numbers and models should make use of them.

5 Using the Model to Evaluate and Build Better Tools

Our "galactic" model can also be used to identify adequate UCD tools for a given project phase. Let us consider, for instance, brainstorming. In this workstyle all values of our model are very low: problem perspective, informal, low detail, unstable (many ideas and many frequent changes), no functionality at all, no need for traceability and people collaborating at same place and at the same time. Paper & Pencil (or Whiteboards) are often used to brainstorm [2]. Figure 5 shows the workstyle model of Paper & Pencil as a thick line. Through our model, we can see that it is a tool almost perfect for brainstorming. However, changes are hard to accomplish using paper/pencil, so the value for stability is high, and we get an indication of the mismatch point between the tool and the desired workstyle. The ideal tool for brainstorming would also need to support fast changes to artifacts, as computer tools do.

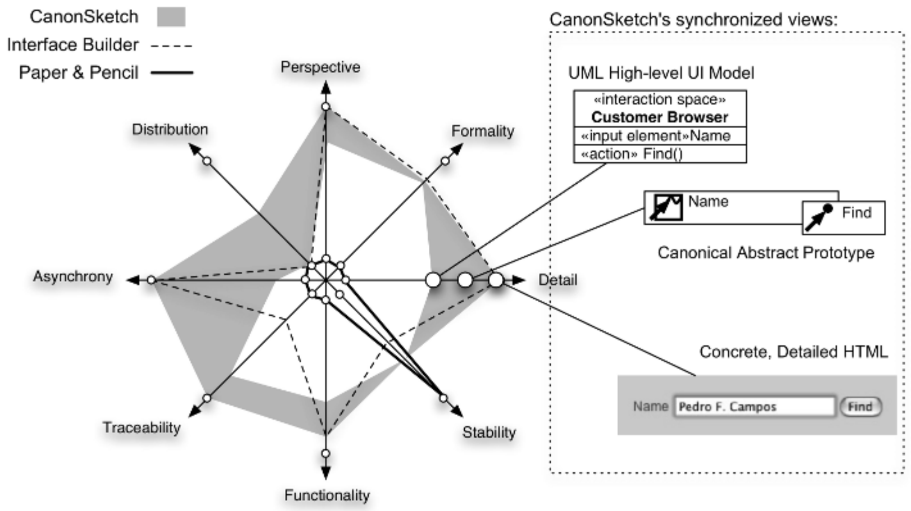

Fig. 5. CanonSketch, Interface Builder and Paper&Pencil under the galactic model

Figure 5 also shows how our model was used to drive the development of a new user-centered tool for designing UI's. CanonSketch [1] is a tool that supports multiple levels of detail by providing the designer three views: UML model of the UI and domain, Canonical Abstract Prototype [3] and HTML concrete prototype (as the right side of Figure 5 exemplifies). The first two views are synchronized and the UML semantic model is used to support traceability. There is also a collaborative version of this tool in which designers can work at the same time on the model and at different places. However, support for distribution is still limited (for instance, there are no awareness mechanisms). Therefore, CanonSketch supports a region in our model, as illustrated in Figure 5. In this way, the tool seamlessly supports designers while switching from high-level abstract views of the UI and low-level concrete realizations [1]. CanonSketch has been tested under a laboratorial setting and has lead to promising results. By contrast, a visual Interface Builder only supports a line in the workstyle model (the dashed line in Figure 5).

Figure 6 illustrates the application of the model to two software modeling tools: ArgoUML and IdeogramicUML. The former is a well-known open-source UML tool and the latter is a commercial tool based on a research project about collaborative software design. ArgoUML has a full-semantic model that allows, for instance, reverse engineering and code generation. IdeogramicUML only supports the syntax of the UML. Thus, ArgoUML is more formal than IdeogramicUML. They both cover part of the perspective, abstraction and modifiability axis; and can even generate some partially functional code. Traceability exists to some extent, since some views are interconnected automatically and there is something like a model navigator in both tools. More differences come in the collaboration-style dimensions. IdeogramicUML uses a sketch recognition language and can be effectively used in electronic whiteboards. There is also a distributed version with awareness mechanisms built in. This way this tool covers a larger region of the model (see Figure 6) than ArgoUML.

Figure 6 also compares the galactic workstyle values for a visual interface builder (VisualBasic) against the popular diagramming tool MS Visio. Visual interface builders became very popular because they support a workstyle in which a functional, concrete prototype can be easily created, thus reducing iteration times. Like ArgoUML, there is no support for collaboration. These tools are limited to a single workstyle (plotted as a single line), so there is no support for abstraction or requirements definition, which is the reason why so many interfaces become rapidly, but poorly designed. On the other hand, Visio supports a wide range of detail, perspective and formality levels and even though it doesn't create functional prototypes, its flexibility in terms of notation quickly became key to its success in rapid prototyping.

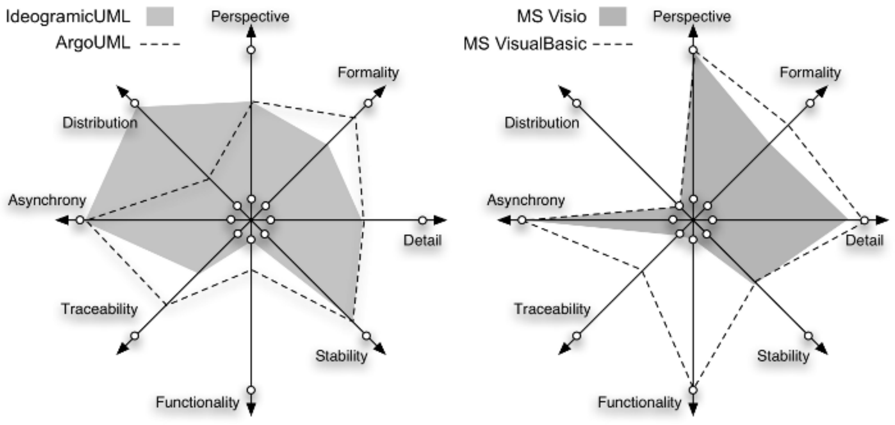

Fig. 6. ArgoUML, IdeogramicUML, VisualBasic and Visio, under the galactic model

These examples show how it is possible to find adequate UCD tool support by applying our model in an easy and intuitive way. It also shows how we can compare and analyze the trade-offs between the dimensions.

More importantly, our model can be used to drive the design of UCD tools, as we have already done with the CanonSketch project [1]. The ideal UCD tool should support the whole space of our "galactic" model as well as shifts between any given workstyle. This could leverage the iterative nature of the UCD process.

6 Conclusions

There is ample room for innovation regarding tool support for UCD processes. Current tools don't fulfill (at least totally) the UI activities of their users: the developers and interaction designers. Support for collaboration and informal communication is even more critical in processes such as UCD. More importantly, supporting transitions in workstyle dimensions can lead to better user-centered tools for user-centered design, given the iterative, evolutionary nature of the process.

Our framework is the first workstyle model tailored to UCD. Current models are too specific and are not expressive enough to be applied more generally to UCD. We expanded them and unified their most significant dimensions in order to achieve a more useful and usable model that could convey more information in a better way. We provided a set of guideline questions that can be used to plot values in the model space. However, our model should be regarded more as an informal discussion tool, rather than a formal method for analyzing workstyles. Contrary to other models, it allows estimating the effort and stage of development of a UCD process by checking the size of the region or line. Its dimensions were specifically designed to allow an easy and intuitive plotting of styles of work. Our model can also be effectively used to (a) choose adequate tool support for a given phase of a project and (b) drive the development of new UCD tools.

We believe that a "glass galaxy" tool supporting these "galactic" workstyle dimensions will cause an impact on the practitioner's productivity and creativity.

References

1. Campos, P. and Nunes, N. J., CanonSketch: a User-Centered Tool for Canonical Abstract Prototyping. In Proceedings of the EHCI/DSV-IS'2004, Hamburg, Germany, 2004.
2. Constantine, L. and Lockwood, L. (1999). Software for Use. A Practical Guide to the Models and Methods of Usage-Centered Design. Addison-Wesley, Reading, MA, pp. 194-195.
3. Constantine, L. (2003). Canonical Abstract Prototypes for Abstract Visual and Interaction Design. In Jorge, J., Nunes, N. and Falcão e Cunha, J. (eds.), Proceedings of DSV-IS'2003 – 10th International Workshop on Design, Specification and Verification of Interactive Systems, LNCS - Lecture Notes in Computer Science. Springer-Verlag.
4. IIvari, J. (1996). Why are CASE Tools Not Used? Communications of the ACM, 39:94-103.
5. Green, T. R. G. & Petre, M. (1996). Usability analysis of visual programming environments: a 'cognitive dimensions' framework. J. Visual Languages and Computing, 7, 131-174.
6. Jarzabek, S. and Huang, R. (2004). The Case for User-Centered CASE Tools. Communications of the ACM, 41(8):93-99.

7. Kemerer, C. F. (1992). How the Learning Curve Affects CASE Tool Adoption. IEEE Software, 9, 23-28.
8. Landay, J. and Myers, B. (2001). Sketching Interfaces: Toward More Human Interface Design. IEEE Computer, pages 56-64.
9. Lumsden, J. (2001). SUIT - A Methodology and Framework for Selection of User Interface Development Tools, Ph.D. Thesis, Department of Computing Science, University of Glasgow, Glasgow.
10. Myers, B., Hudson, S. and Pausch, R. (2000). Past, Present and Future of User Interface Software Tools. ACM Transactions on Computer Human Interaction, 7(1): 3-28.
11. Nunes, N. J., Cunha, J. F. (2001). WISDOM Whitewater Interactive System Development with Object Models, in Mark van Harmelen (Editor), Object-oriented User Interface Design, Addison-Wesley, Object Technology Series.
12. Seffah, A. and Metzker, E. (2004). The Obstacles and Myths of Usability and Software Engineering. Communications of the ACM, 47(12): 71-76.
13. Traeetteberg, H. (2003). Dialog Modeling with Interactors and UML Statecharts - A Hybrid Approach. In Proceedings of DSV-IS 2003, pages 346-361.
14. Wu, J. and Graham, T. C. N. (2004). The Software Design Board: a Tool supporting Workstyle Transitions in Collaborative Software Design. In Proceedings of the EHCI / DSV-IS'2004, Hamburg, Germany.