| Dimension (CLUSTER) | Summary | Range of key examples |
|---|---|---|
| INTERACTION | How do users manifest their ideas, evaluate the result, and generate new ideas in response? | |
| Feedback Loops | How wide are the various gulfs of *execution* and *evaluation* and how are they related? | Immediate Feedback (short) vs. batch mode (long) gulf of evaluation |
| Modes of Interaction | Which sets of feedback loops only occur together? | Setup vs. editing vs. debugging |
| Abstraction Construction | How do we go from abstractions to concrete examples and vice versa? | Programming by Example vs. first principles |
| Implicit vs. Explicit Structure | How much time must users spend accounting for syntax or format errors? | Zero (explicit structure) vs. lots (implicit structure) |
| NOTATION | How are the different textual / visual programming notations related? | |
| Notational Structure | What notations are used to program the system and how are they related? | Notations overlap and need sync vs. complement each other |
| Primary / Secondary Notations | Is one notation more important than others? | Secondary build scripts vs. visual editor and code on equal footing in Flash |
| Expression Geography | Do similar expressions encode similar programs? | Concise yet error-prone vs. explicit yet verbose |
| Uniformity of Notations | Does the notation use a small or a large number of basic concepts? | Lisp S-expressions vs. English-like textual notations |
| Notational Freedom | How easily can the system integrate novel notations that the user has implemented? | Encourages custom notations vs. enforces a single one |
| CONCEPTUAL STRUCTURE | How is meaning constructed? How are internal and external incentives balanced? | |
| Conceptual Integrity vs. Openness | Does the system present as elegantly *designed* or pragmatically *improvised*? | Integrity (Everything is a X) vs. openness (compatible mixtures) |
| Composability | What are the primitives? How can they be combined to achieve novel behaviors? | Sequence, selection, repetition, function abstraction, recursion, logical connectives |
| Convenience | Which wheels do users not need to reinvent? | Small vs. expansive standard libraries |
| Commonality | How much is *common structure* explicitly marked as such? | Common structure is redundantly flattened vs. factored out |
| CUSTOMIZABILITY | Once a program exists in the system, how can it be extended and modified? | |
| Staging of Customization | Must we customize *running* programs differently to *inert* ones? Do these changes last beyond termination? | Source code vs. config files, Developer Tools tab, auto image-based persistence, scripting language |
| Externalizability | Which portions of the system's state can be referenced and transferred to/from it? | None (state is private) vs. all state exposed as human-legible, CSS-like addressing |
| Additive Authoring | How far can the system's behavior be changed by *adding* expressions? | None (requires power to change original) vs. full (anything can be overridden repeatedly) |
| Self-Sustainability | How far can the system's behavior be changed from within? | None (rely on external tools) vs. self-sufficient (contains everything needed) |
| COMPLEXITY | How does the system structure complexity and what level of detail is required? | |
| Factoring of Complexity | What programming details are hidden in reusable components and how? | Domain-specific (more hiding) vs. general-purpose (less hiding) |
| Level of Automation | What part of program logic does not need to be explicitly specified? | Garbage collection (low-tech) vs. Prolog engine (hi-tech) |
| ERRORS | What does the system consider to be an *error*? How are they prevented and handled? | |
| Error Detection | What errors can be detected in which feedback loops, and how? | Human inspection in live coding vs. partial automation in static typing |
| Error Response | How does the system respond when an error is detected? | Does it stop, recover automatically, ignore the error or ask the user how to continue? |
| ADOPTABILITY | How does the system facilitate or obstruct adoption by both individuals and communities? | |
| Learnability | What is the attitude towards the *learning curve* and what is the target audience? | HyperCard for the general public vs. FORTRAN for scientists |
| Sociability | What are the social and economic factors that make the system the way it is? | Cathedral vs. Bazaar model |