Emacs as Design Pattern Learning

Greta Goetz

November 13, 2021

Contents

1 Who is this talk for?

- People interested in thinking about Emacs as a tool sophisticated enough to cater to complex assemblage of tasks, people, activities/outcomes, tools (Markauskaite & Goodyear)
- Learning to learn how to continuously iterate knowledge to changing, complex contexts
- Some software oversimplifies. Emacs both helps users implemenet design pattern learning that can cope with complexity *and* models complex design pattern learning

2 What are design patterns?

- cf. Alexander - design theory in programming (Gabriel) and pedagogy (Goodyear & Retalis)
- Patterns of micro solutions combining method + artefact
- Macro solutions of patterns viewed together (cf. Alexander)
- Allows specialization, customization, extension *and* reuse - Gabriel
- Especially if we're seeking to deal with complexity, it helps extend assemblage of learning components without building from scratch (cf. Goodyear & Retalis 2010)
- Human-centered

3 Why Emacs and design pattern learning?

- Extensibility: Free core (e-quality of opportunity to co-create knowledge cf. Beaty et al.: learning for all)
- Use for different purposes - true of even basic functionalities (language evaluation, buffer cycling, key strokes/basic commands, Org tree outlines, header args/code blocks) support easy re-presentation of material
  - Successful center (feature of design pattern): is made of a center surrounded by a boundary which is itself made of centers - Alexander in Gabriel

4 Why do we care?

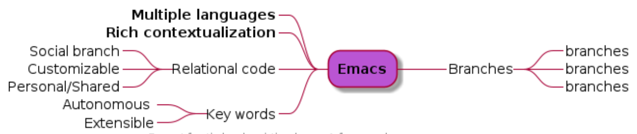
- No pre-fabricated software app silos with limited extensions, stranding work materials
- Assumptions: in UX design and learning. Be your own person(a)
- Practical use can make non-programmers programmers: creative persons as producers and users (Illich), contribute to the evolution of rules (Stiegler)
- Personal toolkit (Stallman)
- Develops heuristics - extensible beyond Emacs

5 Why complex assemblages matter

- Not poor, reductionist contextualization (*a range* of languages can be evaluated on top of Lisp interpreter)
- (Impoverished languages flatten communication)
- Learning to contextualize: First step in learning to learn (Trocmé-Fabre)

6 Mind map to show compex assemblages of Emacs

- (cf. Tony Buzan in Trocmé-Fabre)
- Core; relational codes; key words, cycling (easy key strokes, commands) to bring out ideas (buffer visibility, cycling), branches at periphery, in shared personal configurations



Emacs' frontierless heuristic schema: A framework

- Frontierless heuristic schema
- A **free** system that extends following *our* paths of desire (inits, packages) - but shared tool (Illich) - core at the middle

7 Emacs as a design pattern framework

- Operates through co-constructed knowledge of e-quality (cf. Beaty et al.)
- cf. Gabriel. The "being" of free software: each subsystem (exhibiting behavior in response to requests) - e.g. Magit - is a center, but being part of Emacs, we have a system made up of other systems: "communicating components that work together to provide a comprehensive set of capabilities that can be customized, specialized and extended to provide more or slightly different capabilities"

8 Personal customizations

- Emacs as general computing
- 'Wise' use of computers (Crichton)
- Everyone's Emacs is their own
- Can be used easily by anybody as often or seldom as they want for the *purpose* chosen, shaped according to taste (Illich)

9 Using - basic - Emacs design patterns for learning

- (cf. Guo et al.)
- Modularity (e.g. Org tree outlines, header arguments) that supports representation to meet specifications of changing contexts (shorter lecture segments, different deliveries)
- *Topoi* accessibility

- Helped by languages (e.g. PlantUML), packages (org-ref), workflow possibilities (Sacha's completing sketches)
- Can learn how to learn by following the traces left by others in community

10 Social capacities

- Grammar of interaction (Gaume in Andler & Guerry)
- Co-individuation (meaning known and shared by other individuals, Stiegler)
- Without the social milieu, the technical milieu inevitably becomes a negative externality (knowledge automatization is a closed, self-referential system that turns users into servants) (Stiegler)
- Take care of neighbors & excel at using the best available tool (cf. Illich)
  - The shared core evolves (just like we configure/program while using): as a model of learning

11 Emacs community design pattern: Cognitive democracy

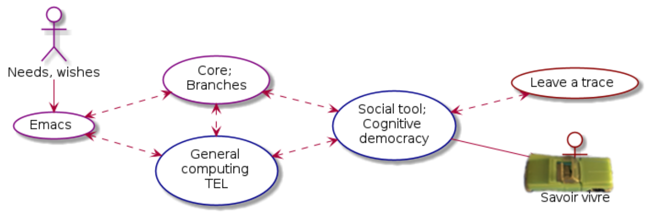
- First, there is a community (e.g. Sacha, PlanetEmacsLife)
- Different/competing views
- Morin: nourished by regulated antagonisms
- Gabriel's centers of centers, the "being" of free software allows this range of extensibility

12 The Emacs center of centers: expanding, relational, free

- Only in some systems does the "being" emerge, the framework that can be used and reused and which gives systems and objects their spirit - Gabriel
- Values the value of the freedom to create, use, and share (Stiegler) - community spirit
- Autonomous designer mindset: design pattern iteration (Gabriel)
- Not 'flattened': permits ongoing learning, reassembling contexts; adaptable design pattern extensibility
- Helps create circumstances where learning is coherent with what is valued in the rest of life: pleasure, growth, transformation (Goodyear et al.)

13 Thank you

Thank you to the developers, maintainers, contributors, and community for championing our freedom to co-individuate complex design patterns the way we want to, so we, too, can leave original traces - if we want to!


Needs, wishes
Emacs
Core; Branches
General computing TEL
Social tool; Cognitive democracy
Leave a trace
Savoir vivre