

Lightweight & Flexible Requirements Modeling

Assignment 6 - Task C

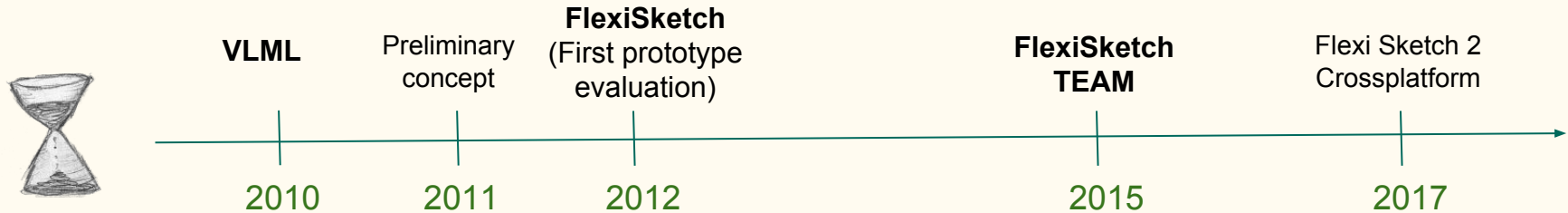
Justin Sosnoski & Maria Herdt

Overview of papers

Very Lightweight Requirements Modeling - Glinz 2010

FlexiSketch - Wüest 2012

FlexiSketch Team: Collaborative Requirements Elicitation - Wüest 2015



Overview of papers - Notes

VLRM (2010) - a “position” paper covering how a very lightweight model may be quickly “sketched” and converted with (potentially) minimal modeling experience, and presented immediately to SHs for feedback and re-design. Proposes a new lightweight modeling language VLML

FlexiSketch (2012) - this is a requirements modeling app that builds on the 2010 vision of Glinz of a VL requirements model, and allows for a digital model to be built “in the field”, with little restrictions on the structure (“language”) used

FlexiSketch Team (2015) - extends the flexisketch platform previously built in 2012 with multiple app interconnections, allowing for collaborative work between team members and also supporting larger display options (from a single tablet -> multiple tablets + large team display screens).

Need for lightweight & flexible requirements modeling

- Requirements at early SE stages are narrative and pictorial
- Heavyweight model languages (e.g. UML) do not fit this needs

Proposal to create and use VLML [Very Lightweight Modeling Language] as alternative to textual and pictorial specifications



Lightweight:

- small language
- easily integrates with nat. language and helps to structure it
- simple modeling constructs as alternative to text for structure, relationships, influence, flow



Flexible:

- Users can sketch any informal models or diagrams
- Syntax and semantics can be assigned on the fly (language/notation-independent)

VLML model & tool support

Model: Set of objects that may be specialized by modifiers and can have relations among each others

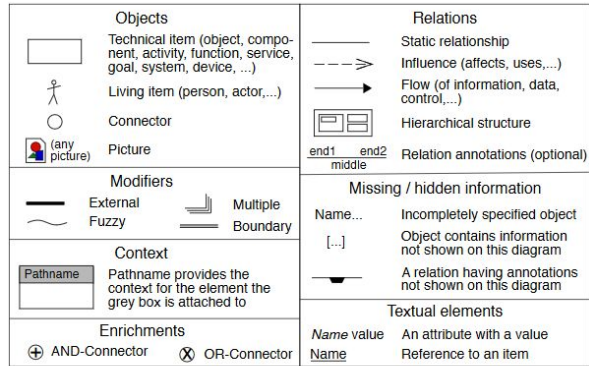
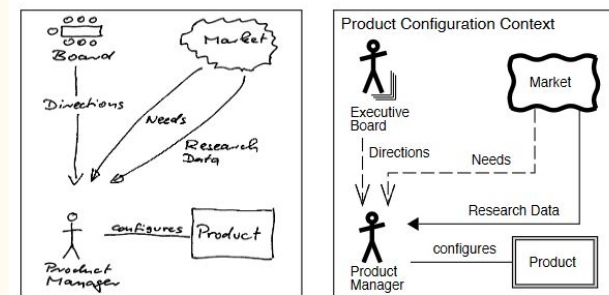


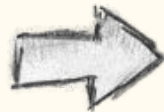
Figure 1. The visual syntax of a sample VLML



Automatic inference of items such as Product Manager should be possible, while others (e.g., recognizing Board as a group of actors) will require human guidance.

Figure 2. Converting sketches into VLML models

- Hierarchical structure & powerful abstraction capabilities of VLML can be exploited by tools
- Conversion from sketches to models can be also tool-supported



FlexiSketch

VLML model & tool support - Notes

Model is very intuitive and easy to understand even without previous knowledge.

Consists of

1. Objects: can be a technical item, living item(person or actor), connector or a customized picture imported
2. Objects can be specialized by modifiers (e.g. Multiple, fuzzy, external)
3. Relations among the objects (static, flow of control or influence)

Picture 2 shows the process of converting sketches into VLML models (some inference happens automatically, some conversion needs human interaction)

Hierarchical structure & powerful abstraction can be exploited by tools and the automatic conversion can be also tool supported



FlexiSketch



Mobile tool (Android) for model-based sketching of free-form diagrams. Single-user version.

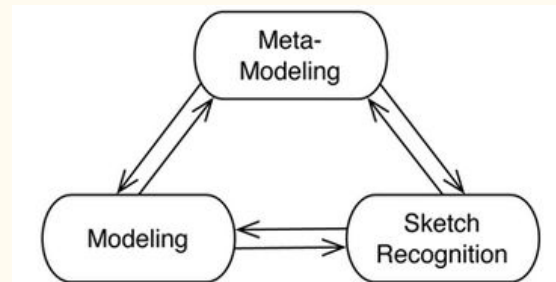
Concept: 2011

Prototype evaluation: 2012

- Alternative to paper&pencil or whiteboards in early SE phases
- Creation of free-form sketches and arbitrary node-edge diagrams
- Allows assigning syntax and semantics to sketched elements
- Supports further transformation into classic semi-formal models (e.g. class diagram)

Approach: switching between 3 modes

1. **Modeling**: create, augment, modify sketch
2. **Metamodeling**: create elements based on user's annotations
3. **Sketch recognition** based on information from meta-model





FlexiSketch - Notes



Approach Lets user arbitrary switch between 3 work modes

Modeling mode: user creates and modifies sketches

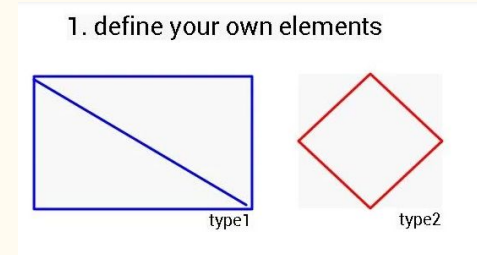
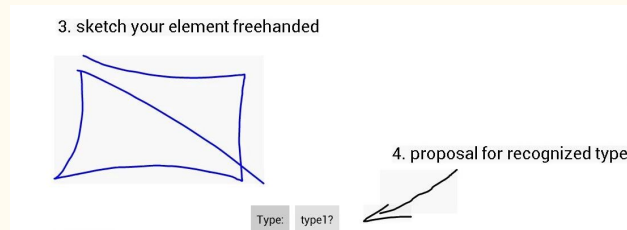
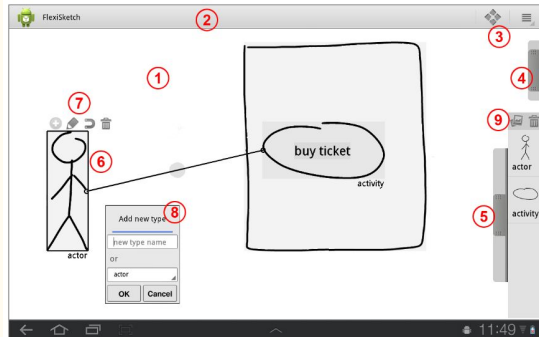
Metamodeling: user can annotate sketch and tool creates metamodel based on annotations

Sketch recognition: tool transforms sketched elements into semi formal model elements by recognizing and interpreting sketches based on information currently available in the metamodel.

To avoid distracting the user, the switch between these three modes happens implicitly

FlexiSketch: main features

- Sketching possible with fingers or with stylus pen
- **Object recognition:** identify completed elements
- **Sketch recognition algorithm:** compare a new element to library and identify similar/matching $\Rightarrow$ Trainable - learns from user input
- Automatically **builds metamodel** according to the sketch and **type library** definitions
- **Type library management** (list of user-defined types from the sketch + visual representation)



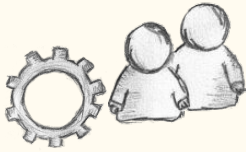


Benefits

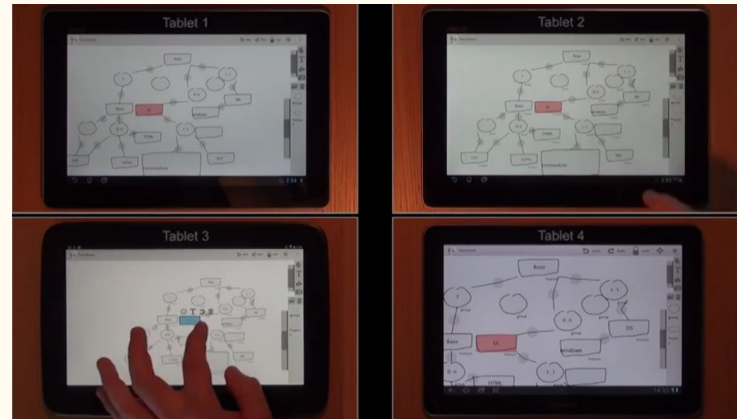
- Informal sketches for early SE phases: brainstorming, elicitation, idea generation
- Reduced “media break”
 - Note: with pen and paper or whiteboards, contributors typically take photos of sketches for documentation. These photos are not editable so they must then be manually converted to digital models. Each of these “transformations” is a media break.
- Fewer opportunities for wrong interpretations → less errors
- Saves time: no need to re-model gathered information manually from scratch
- Lets the user define and annotate own diagram symbols and not just reuse predefined → unique niche tool
- Images can be imported into sketch and be treated as symbols
- Simplified version of semi-formal diagrams, something that stakeholders with different backgrounds understand

FlexiSketch Team

Add-ons:



- Synchronous, multi-display collaborative sketching
- Focus on co-located settings
- Collaborative definition of notation on the fly, reusable in later SE sessions
- Public/Private work modes





Limitations & Drawbacks

- Reliant on hardware accessibility: each user needs a tablet to contribute, and groups work best with a team display screen to show “big picture”
- Hardware/software imprecision → less “natural” feeling
 - Screen thickness or software lag may cause a “disconnect” between writing implement and sketch visualization
 - Difficult to select/delete small elements
- Smaller surface area
- Links/relations by line only
- Less group interaction and communication than with P&P or whiteboards
 - Note: FlexiSketch Team addressed this. However, in the FS Team trial the interaction and communication was primarily focused on defining notation and with “status checks,” rather than problem solving.

FlexiSketch 2 (Future)

Add-ons:

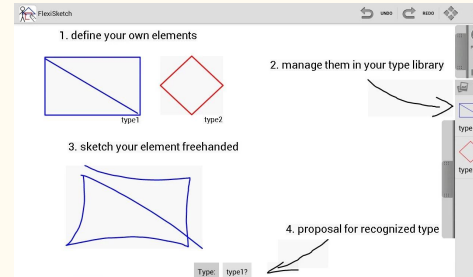
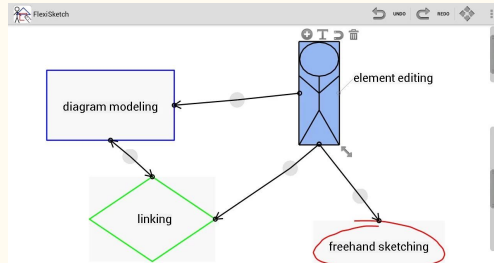
- Crossplatform (iOS, Android, Mac, Windows)
- Integration of text documents





Discussion

- The FlexiSketch authors pointed out that mobile devices are more pervasive than whiteboards, and imply that the accessibility of a mobile app could be sufficient to replace traditional whiteboards. However, the studies strictly used **tablets** (not smartphones). Given that the published app is only intended for tablets, how would this affect location-independence?
 - If it is available and practical (usable) on a smartphone, how would this change your response?





Discussion

- FlexiSketch Team endeavoured to increase team collaboration and communication in the app. In what **other** ways can the limitations of the app be overcome?



Limitations

- Reliant on hardware accessibility: each user needs a tablet to contribute, and groups work best with a team display screen to show “big picture”
- Hardware/software imprecision → less “natural” feeling
 - Screen thickness or software lag may cause a “disconnect” between writing implement and sketch visualization
 - Difficult to select/delete small elements
- Smaller surface area
- Links/relations by line only
- Less group interaction and communication than with P&P or whiteboards