

UI Design Example: Babble

James O'Brien

jaimzo@gmail.com

This document is an overview of the Babble project I created at MSRC. I'll focus on the UI for Babble, highlighting the reasoning behind specific design decisions. The project was written in C# using .NET 1.2, utilizing GDI+ for the text and graphics rendering and Pastry for the P2P communication. I did all the programming and graphic design.

Joyce

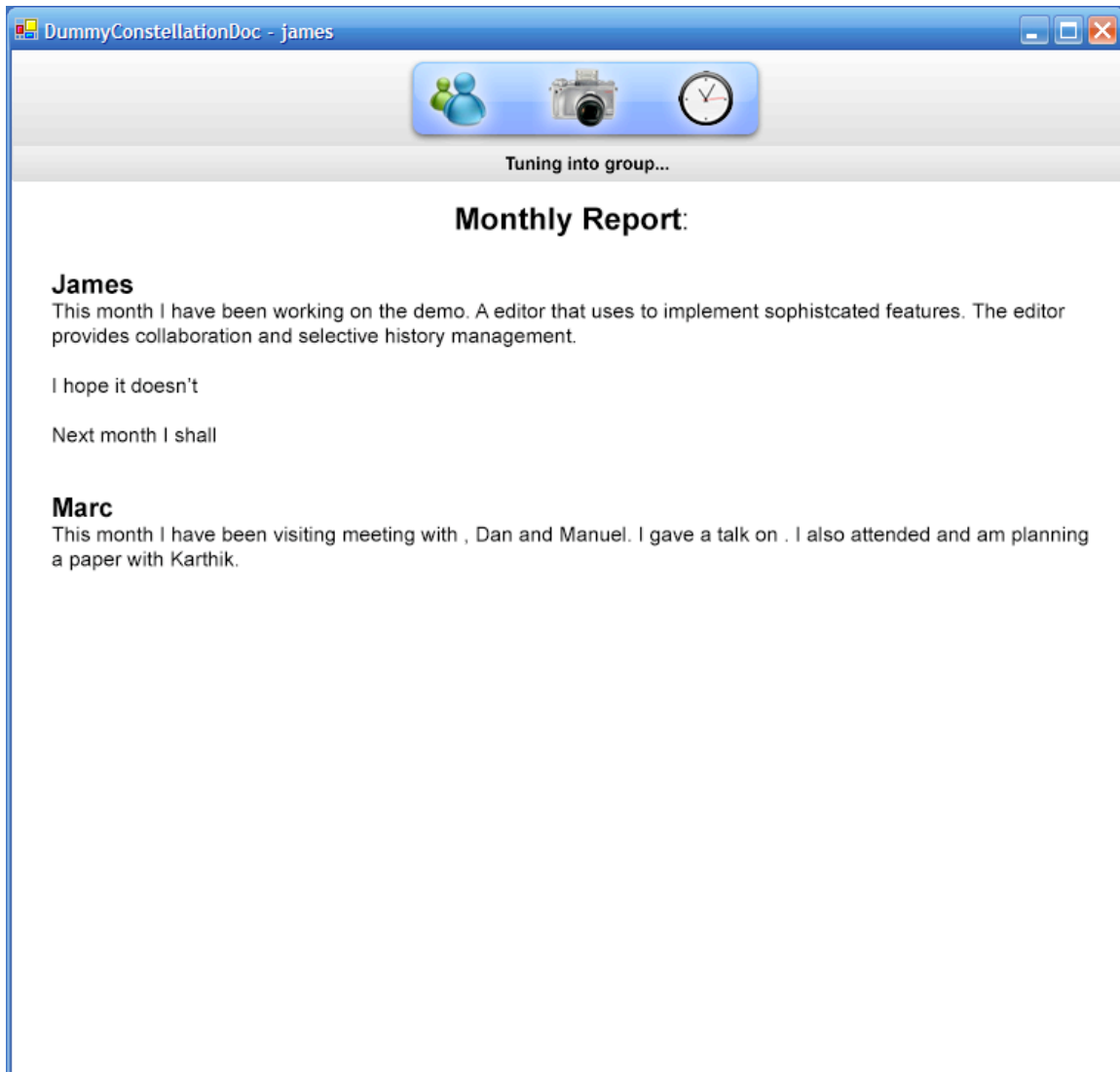
Joyce was a project that grew out of the IceCube reconciliation engine previously developed at MSRC. IceCube allowed applications to synchronize offline changes by modeling application activity using constraints describing semantic invariants between operations. When synchronizing, IceCube would walk a graph of these operations and constraints to extract the largest, semantically consistent subgraph – which would then be used to generate a *globally* consistent state.

Joyce applied the principles of IceCube to *nomadic, collaborative* applications. That is, multi-synchronous, collaborative applications wherein the set of collaborators, the connectivity between them, and the synchrony of collaboration may change at any time. Some advantages of the Joyce platform were fluid, lossless changes between online and offline modes and rich, group-wide history editing (selective undo/redo).

Babble

To 'sell' the Joyce technology to product groups we needed an application that served as a demo, a prototyping test-bed and, potentially, a source of code transfer to specific product groups. In particular, I identified the group working on collaboration in Vista as a group that had specific problems that were solved by Joyce.

To highlight and sell Joyce I picked a familiar application – a document editor – and applied the Joyce technology to it. The idea behind picking such a familiar application was that the application itself would 'fade away' and allow people to more easily grasp the benefits of the Joyce framework. With this in mind, the key to designing the Babble interface was to create a *sparse* interface that was instantly familiar but clearly highlighted the contributions of Joyce:



The *start-of-day* interface (above) is plain, functional and familiar. Aside from the text editor there are just three controls corresponding to the three broad aims of Joyce. For demo purposes the editor was pre-populated with text.

Design point

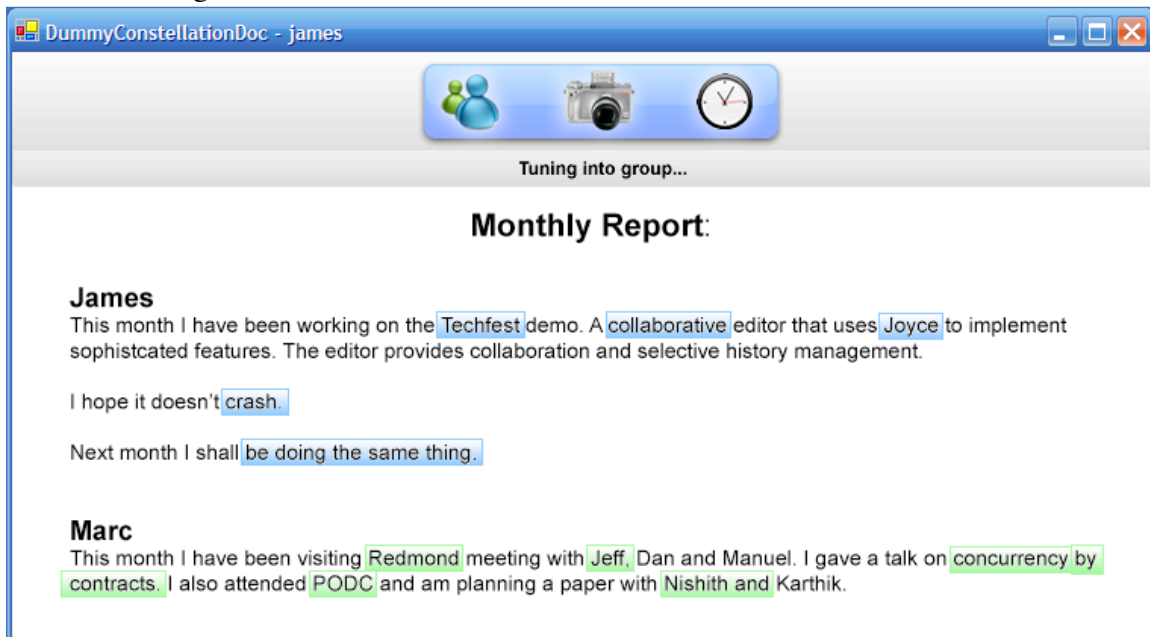
As with all research demos, the 'interface' used in the demo is not necessarily the most *usable* interface – you are creating the interface to sell the research rather than for a customer to use every working day. In Babble, the Joyce framework was already very robust and Babble was designed to be flexible enough that the demo interface could be swapped-out for a more product-oriented interface.

Editing



The controls at the top of the display introduced the advanced functionality that Joyce provided. These were the only controls displayed at the 'top level' and were designed not only to simplify the interface but also to guide the user's

attention through the demo.



This shot shows the editor during a two-user collaborative session. Note that the contributions of the users are highlighted within the content; hovering over an edit displayed more information about the origin of the edit:

. A collaborative editor that uses Joyce to
n a **James** collaborative (inserted) history management.

This is known as *tagged editing* – a capability sought after by the collaboration group at the time.

ort: **February**
James
inserted *January*
Marc
inserted *February*

Conflicts were also highlighted in-place and this time the user could select the edit to apply using the overlay graphic. *Note:* by following the user's locus of attention (the content) and not forcing him to visit auxiliary panels to track changes and resolve conflicts, we work *with* the users flow of thought and provide a better experience.

Contributors

The controls at the top of the display show

and hide transient panels to access Joyce functionality. For example, this panel shows the collaborative group working on the document:

DummyConstellationDoc - james

Tuning into group...

Monthly Report:

James
This month I have been working on the Techfest demo. A collaborative editor that uses Joyce to implement sophisticated features. The editor provides collaboration and selective history management.

I hope it doesn't crash.

Next month I shall be doing the same thing.

Marc
This month I have been visiting Redmond meeting with Jeff, Dan and Manuel. I gave a talk on concurrency by contracts. I also attended PODC and am planning a paper with Nishith and Karthik.

Participants

James active block	Marc active block	Ant blocked unblock
--------------------------	-------------------------	---------------------------

The user can *block* a collaborator – whereupon the changes from that collaborator are removed from the content (but *not* lost – they, and any other concurrent edits by the collaborator can be re-introduced to the content by unblocking the contributor).

History

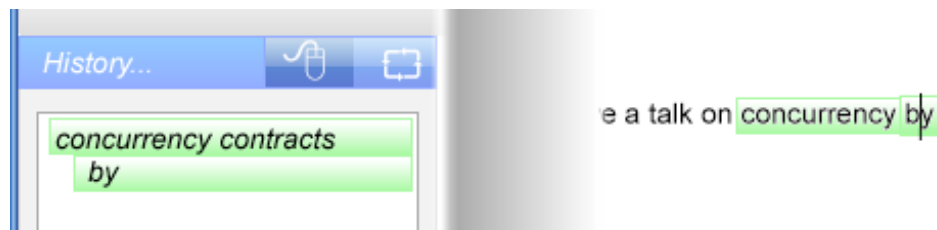
The right-most control displays a *history editor*, which is essentially a selective undo/redo facility (as opposed to the stack-like undo/redo found in current applications.)



The user can undo a specific edit using the control at the top-right of the panel. The undo operation propagates *only to semantically dependent* operations and the content is updated instantly. Undone actions are not lost and are still displayed in the history: for example, see the “Palo Alto and” operation in the above shot. These actions can be selected and redone.

Design point: Local History

This display can get very confusing with large documents and/or lots of collaborators. I introduced a *local history* mechanism that filtered the history display only to actions that affect content beneath the current caret position.

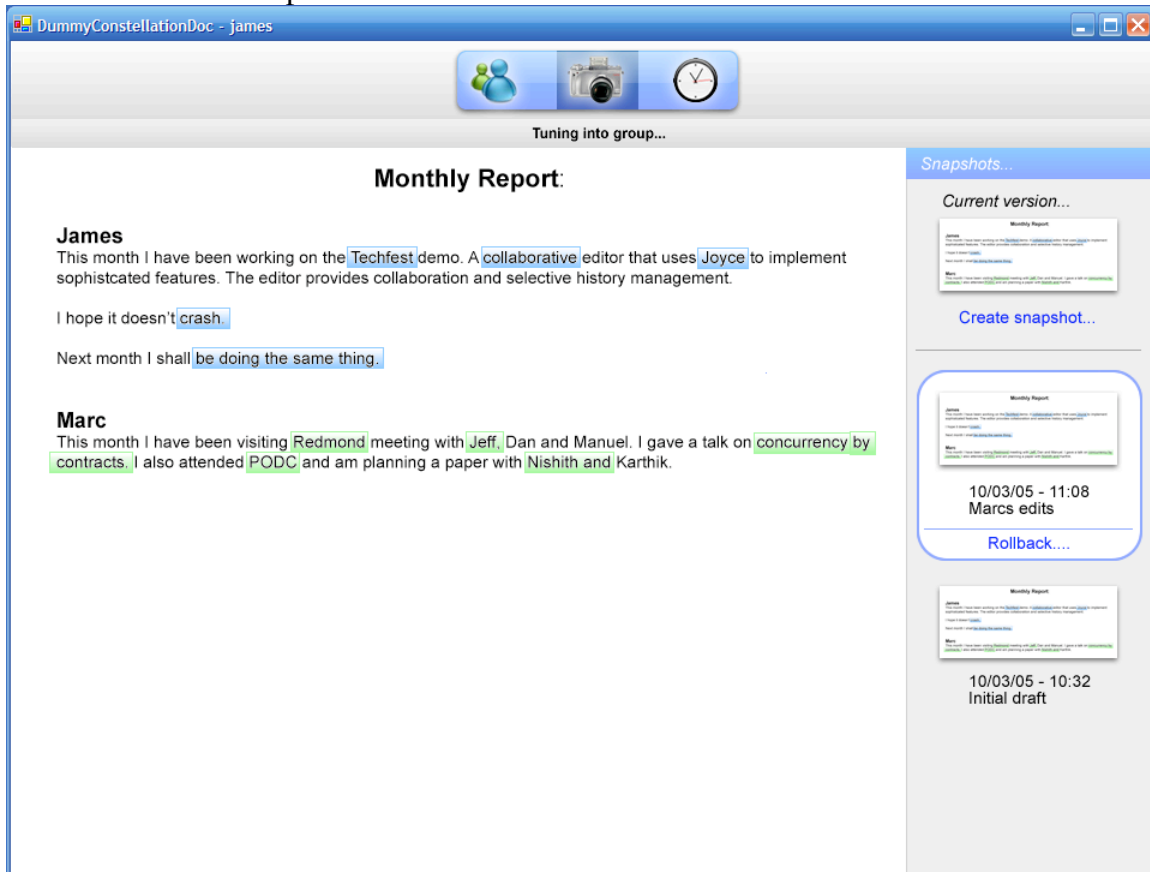


This proved to be a far better way to visualize and navigate history in a ‘busy’ document.

Snapshots

The user never explicitly saves a Babble document – Joyce takes care of persistence transparently. However, a user may snapshot a particular point in a document’s lifecycle, either to roll back to that point later or to distribute the snapshot to other teams – the collaboration and Office groups in particular identified this as a common customer request.

This is the snapshot interface in Babble:



Each snapshot is visually represented by a thumbnail of the content at the point the snapshot was made. The thumbnail at the top of the list represents the current state of the document and updates in real-time according to edits in the content.

To create a snapshot, the user triggers the *Create snapshot...* button. A 'snapshot' of the live thumbnail is made to correspond to the snapshot of the content and this is added to the snapshots list. To roll back to a snapshot the user selects the snapshot's thumbnail, this reveals a *Rollback...* button which restores the content to the state at that snapshot. All updates to the content happen instantly.