The Making Of User-Interface Designer A Proprietary DSM Tool

ドメイン・スペシフィック・モデリング手法を用いた
ユーザインタフェース・デザインツールの開発

Laurent Safa ローラン・サファ
松下電工㈱・EMIT ミドルウエア研究所
EMIT Middleware Laboratory Matsushita Electric Works, Ltd. 1048, Kadoma, Osaka 571-8686, Japan +81-6-6908-6752 safa at mail dot mew dot co dot jp

要約

本資料では、ドメイン・スペシフィックなモデルからコードを自動生成させることにより、従来型開発の5 倍の生産性向上を可能とする、ユーザインタフェース・デザインツールを開発した事例を紹介する。はじめに、開発にいたる背景と、ドメイン・スペシフィック・モデリング手法(DSM)に取り組んだ理由に関して紹介し、次に、仮想モデリング、および動的モデリングに重点を置いた、ドメイン・スペシフィックな記法に関して説明する。最後に、ツールの適用とメンテナンスに関しての考察で締めくくる。

はじめに

今日のように、あらゆるものがデジタル化され、接続されるようになることで、ホームコントローラにおいても、エネルギー削減機能 [1]、侵入検知、地震警告など、より多くの機能が提供されている。これら新しい機能には、メニューをナビゲーションし、データを視覚化し、システムを設定できる徹底したユーザインターフェイスが必要である。また製品システムは、複数のグループで開発されたデバイスで構成されるため、統合されるものが増えるに従い、ソフトウエアの複雑さも増加している。

弊社では、アプリケーションフレームワークを構築して、最適なコード実装と、選り抜きのコンポーネントの再利用を実現することで、これら課題に対処することにした。ただ、ユーザインターフェイスに対する市場要求を調査しながらであったため、フレームワークそのものは、開発者の創造性を阻害しないようにするため、意図的に制限することとした。

結果、フレームワークのプログラミングインターフェイスはドメイン・スペシフィックな設定ファイルと、アプリケーションのスケルトン生成とし、追加のビジネスロジック、ビジュアル・デザインは、プログラマーに残された。

課題

アプリケーションフレームワークは、デバイス間の振舞いの不一致を抑止するのに効果的があることが証明され、システムのインテグレーションが機械的にできるようになった。しかしながら、開発者から、フレームワークを用いることに直接関連した、新しい問題が報告されることとなった。

－フレームワーク固有のAPIに対するドキュメント、及び良いツールの欠如
－エラーの要因となりやすい、複数ファイル間における名前のマッピング
－生成されるスケルトン内への、ビジネスロジックの実装は管理・制御できない
（結果エラーの要因となり、自動化による質の向上という目的に反してしまう）
－ソフトウエア構築のために、追加の手続きが必要となる

フレームワークを用いて複数の製品を開発した後に、組み込みユーザインターフェイスの必要性を、良く理解することができました。そしてまた、視覚的なモデリングツールを追加することで、フレームワークを拡張させることにしました。このツールにより、完全なコードを生成させて、体系的にヒューマンエラーを削減させることを目指しました。

調査・検討

既存のユーザインターフェイスより、ページ内の視覚的デザインパターン（レイアウト、プロポーション、タイプ、情報量など）、相互接続されるページのパターン、そしてアプリケーションソースコードそのものを、浮き彫りにすることができた。これら全てのパターンは、デザイングループにより定義された、ビジネス固有のガイドラインから起因している。

また同様に、要求仕様書からも社内独自の表記なども確認できた。この表記は、矢印、多様な幅の線、ナビゲーションに関わる色、特別な振舞いを指定するアノテーションなど、ページの現実的なスナップなるものであった。

最終的に、既存ユーザインターフェイスの 60%～80%が 5 つの最も多く使用されているページパターンで定義されている事がわかった。

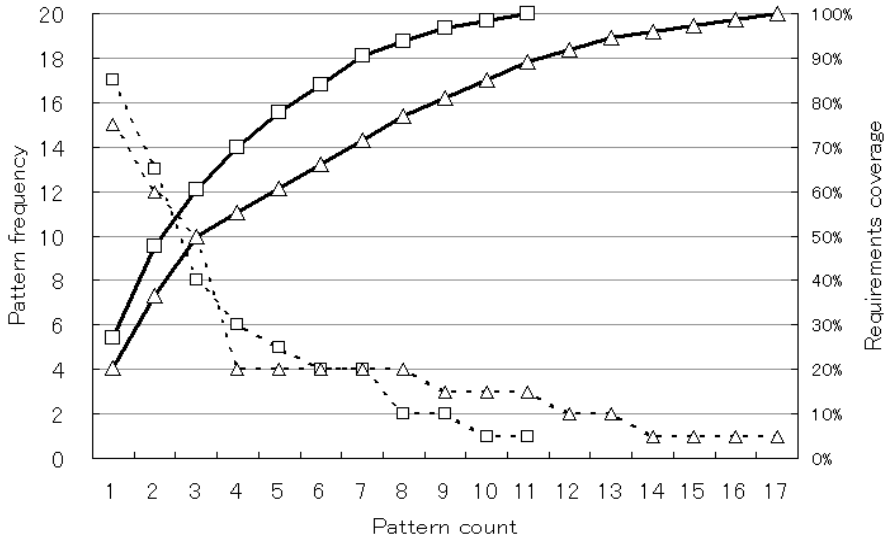

Fig. 1. パターンと要求のカバレッジ

調査の結果として、ドメインは視覚的なモデリングに有利であると判断し、ツール開発に取組むことにした。

4. 形式化

既存の開発ガイドライン、パターンの用法は、ページの標準として、一定の形式を設けた。

例えば、メニューページは、飾りのアイコンからテキスト情報、ナビゲーションボタンなど 13 の部品からなる標準パターンで形式化した。

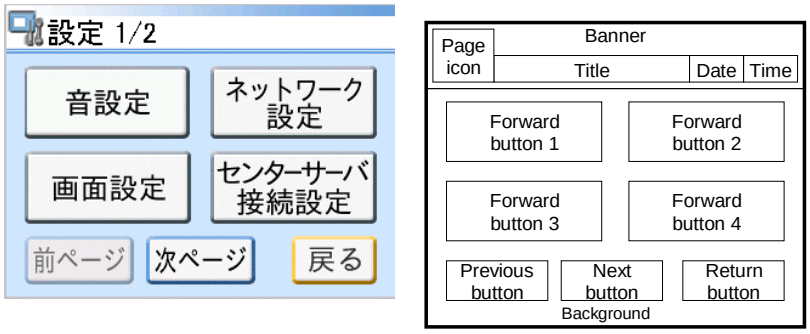

Fig. 2. メニューページのパターン

他、反復するパターンとしては、配下の設備とのやりとりを含む、設定ページ。

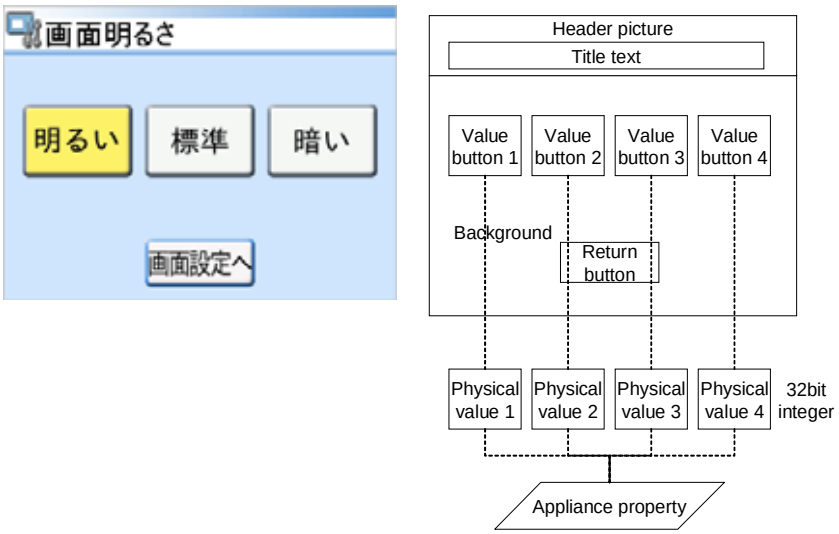

Fig. 3. 設定ページのパターン

「前へ」や、「次へ」ボタンを使ってメニューページ間を分割するような振舞いもまた、形式化した。

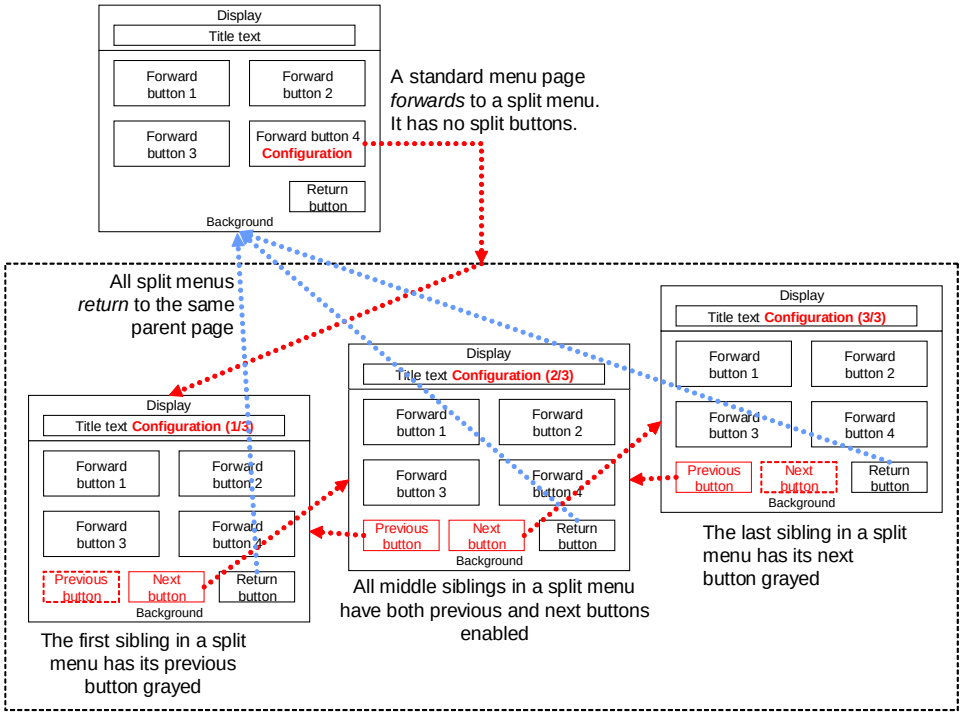

Fig. 4. メニューの振舞いのパターン

ページの標準と、振舞いの標準を明確にできたことで、ビジュアルモデルから完全な製品生成を実現する準備が整った。

選択した技術

少ないリスクによる新製品開発、既存の確立した手法などが優先される為、ツールの開発に割り当てることのできるリソースは限られていた。このような状況に対処すべく、最小限のコーディングで安定したツールを構築できる、モデリングツールのためのツールキットを探すことにした。その、選択基準は以下の通り：

－プログラミング無しにメタモデルと、視覚的なエディタを定義できること。さらに最小限のリソースでまかなう為に、自動化によりアプリケーションプログラミングエラーを削減させることを証明するためにも、コーディングを最小限に止めるという基準は重要であった。
－アプリケーションモデルをスキャンして、ソースコード、makeファイル、各種テキスト成果物を生成させる強力な機能性
－ツールを提供される側と相互作業を容易にする機能
－サポートやトレーニング等のサービスが受けられること
－表記設定、デザイン、自動化など当面の目標に集中できるような、安定した手法・ツール

MetaEdit+® で提供される、GOPPRR手法[2] と、MERL scripting language を結果として採用した。

4. ユーザインターフェイス・デザインツール

GOPPRR と MERLによるツール構築に関しては、別の文献に詳細がある[3]。その為、この内容に関してはスキップし、我々のツールに関する主要な部分に関して紹介する。

6.1 オンサイトで、構成可能なモデリングツール

以下の図は、ユーザインタフェース・デザインツールの代表的な作業セッション。

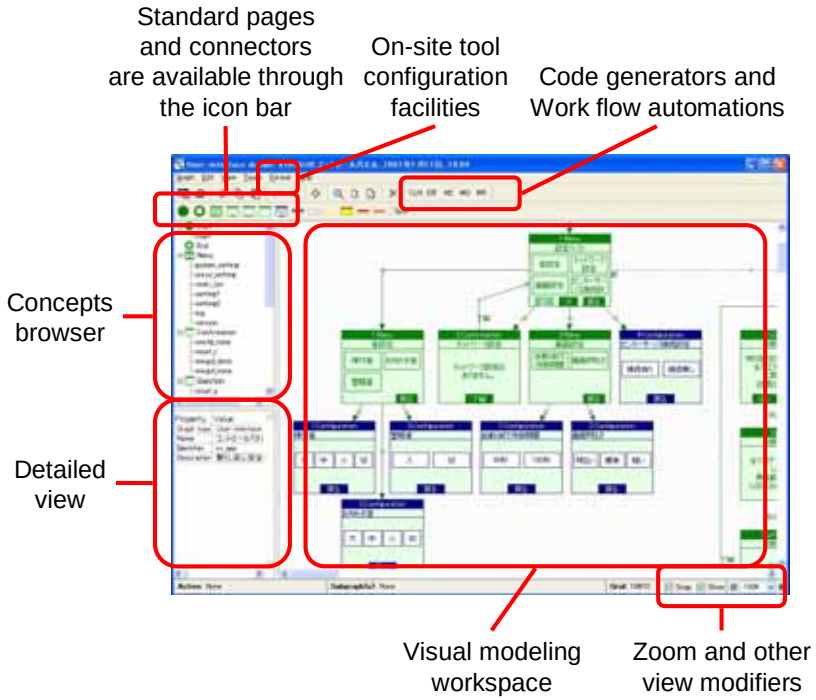

Fig. 5.ユーザインタフェース・デザインツール

視覚的なエディタに良くある、コンセプトや概念のブラウジング、アイコンバーなどに加え、MetaEdit+® では、オンサイトでツールを構成できるようになっている。そのためメタモデルや、視覚的な表記、ジェネレータなどの改修は、エンドユーザで開発に用いている真っ最中でも可能であり、ユーザの提案を、その場で一緒に確認することが、面倒なプログラミング言語や時間の掛かるコンパイル無しに行える。簡潔・容易に変更できるということで、安心感を持ってモデリングツールを用いることの、相当な助けとなり得た。

6.2 サブモデルの包括

サブモデルを包括できるように、視覚的なステートメントの記述を提供した。複雑さを上手く取り扱い、音の設定や、ファームウエアのアップグレードなどの、スタンダードモデルの再利用を円滑にすることが、その目的。これは、従来どおりの、要求セクションごとに、機能が関連するページを分類する手法と、合っていた。各サブモデルは、一要求セクションに該当し、包括される各ステートメントは、要求仕様書の参照/再利用されるステートメントに相当している。

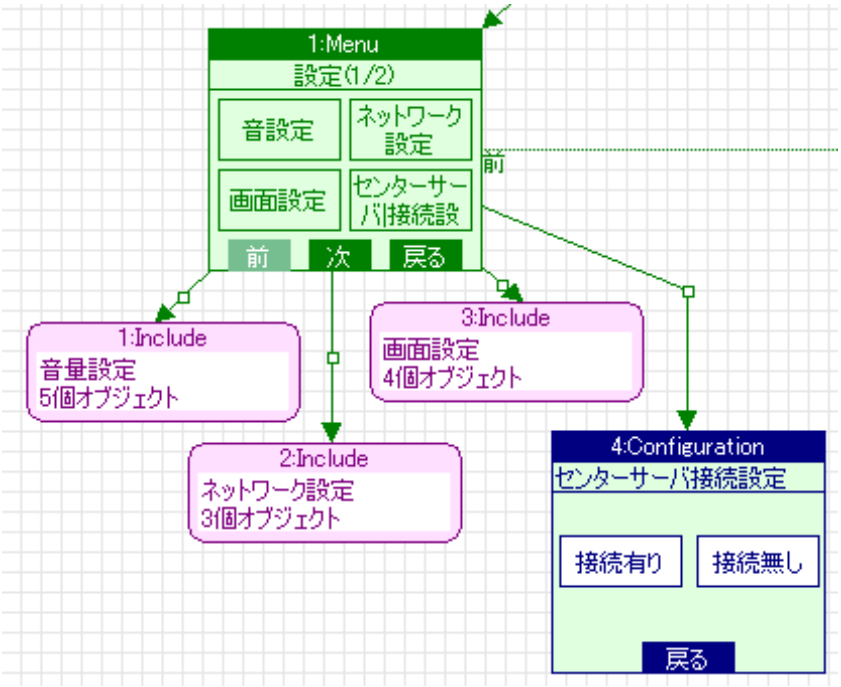

Fig. 6. モデルの包括

6.3 仮想－実在的モデリング

調査を進めていくうちに気付いたことは、規定されるユーザインターフェイスは、現実の製品ページのスナップショットであること。モデルを最終製品に、できるだけ沿うようにすることは、製品構築に関わる関係者が多様であることから、必須となっていた。この必要条件を遵守し、視覚的なシンボルは、製品ページのビジュアル・デザインを、そのまま模倣することにした。例えば、メニューページのシンボル（Fig.7）は、フォワードページ数、“前“ や ”次“ などのボタンにより、分割されたメニューの連携有無、親ページの有無に従い変化する。

Fig. 7. 仮想－実在的メニュー・シンボル

同様に、コンフィギュレーションページのシンボルは（Fig. 8）、選択肢数に応じてボタンのサイズが、忠実に変更される。

Fig. 8. 仮想－実在的コンフィグレーション・シンボル

仮想－実在的シンボルは、最終製品のページに厳密な複製ではないが、ツールユーザにとって、貴重な情報として用いられ、(Fig. 7)のようなメニューのシンボルをパッと見ることで、右が詰っていても、左端メニューに、追加のボタン領域があることが判るようになる。

6.4 動的なモデリング

シンボルは、モデル内のコンテクストを基に、動的に変更され、モデルは編集行為を即座に反映する。Figure 9 は、２ページ間がリンクされる前と、された後のモデル。図の通り、右側のモデルでは、メニュー・シンボルに、追加のボタンが自動的に加わりコンフィグレーション・ページを呼び出せるようになり、そのコンフィグレーション・ページには、追加の“戻る”ボタンが、自動的に付加される。これらページ間のリンクを外すと、最初の状態（左）に戻るようになっている。

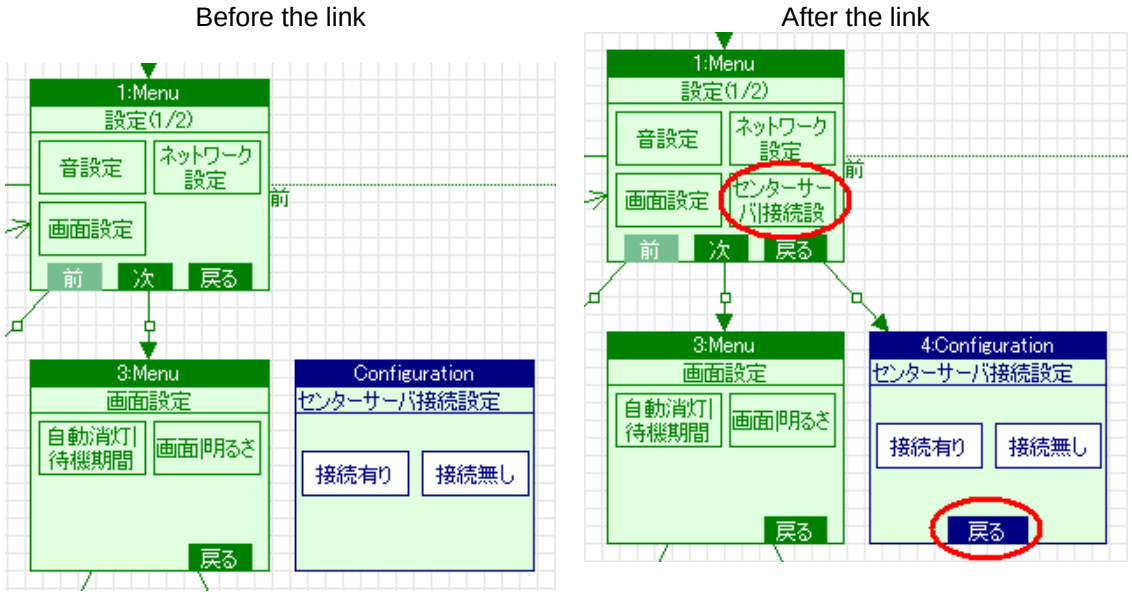

Fig. 9. Dynamic symbols

6.5 コード生成と生産性の向上

ツールの最初のリリースでは、５つの最も頻繁に用いられるページ（*menu*, *configuration*, *question*, *confirmation*, *splash*）をサポートし、モデル化し、60〜80%に及ぶ代表的なアプリケーションが生成出来るようになった

様々なパターンに及ぶ、同等レベルのコードの複雑さを想定すると、これはソフトウエアデザイン、実装、構成フェーズの生産性を3倍から5倍向上させたことになる。ドキュメンテーション、製品テストなど、より一層の効果が期待されるが、まだ十分には測定されてはいない。
DSM を手段として、極めて創造的な製品デザインに対して、改善の見込みは期待していなかったが、当初の結果から、デザイン生成、デザインとコードの統合化も、同様に 3 倍から 5 倍の生産性向上が得られた。

適用

7.1 ツール費用/ 効果の比率

各ページの標準のビジュアル表記、コード生成機能、メタモデル環境実装などに、３人/日 を費やした。１ページ辺りの開発に 0.5人/日かかることを想定すると、その6倍、あるいはそれ以上でもメタモデリングの価値があると考えられる。

7.2　エスケープ・セマンティックの必要性

反復して用いられるパターンに加えて、使用頻度の少ない、ツールで自動化するまでもない、多くのページが存在する。これらは“ようこそ”ページ(Fig. 10)や、製品固有のデータ表示ページなどである。

Fig. 10. Low-frequency patterns happen to contain high-value element

この問題は、ツール供給と実践のギャップにも該当する[4]。この件に対処する為には、新しいタイプのページに対する、セミ・コントロールな、自由なモデリングを提供する、エスケープ・セマンティックが必要である。

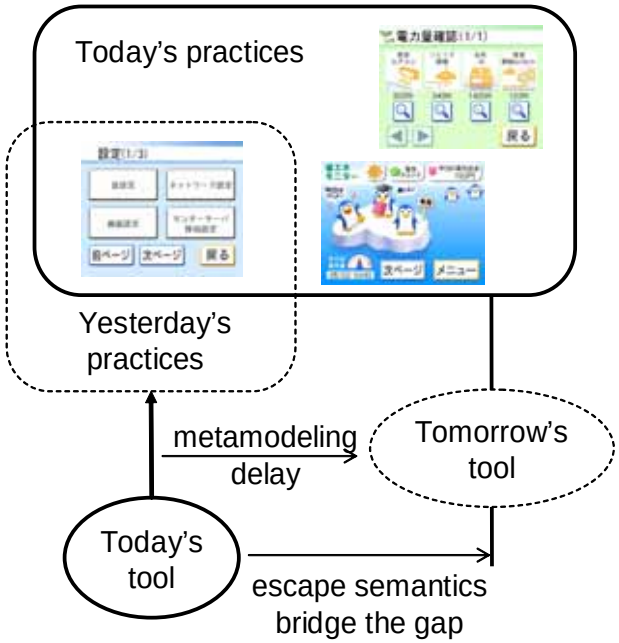

Fig. 11. メタモデル環境の遅延

7.3 エスケープ・セマンティックの例：QVGA Joker

未定義（ジョーカー）シンボルを用いて、エスケープ・セマンティックを提供した。目的は、モデリング担当者が、ツールではまだサポートされていない、新たな製品機能を定義できるようにし、それらをメニューやコンフィグレーションなどの現状サポートされているページのパターンに、統合できることである。Fig. 12 では、シンプルな未定義（ジョーカー）のテンプレートを紹介している。（色により、ジョーカーを識別。赤は、メニューから。緑はコンフィグレーションから）

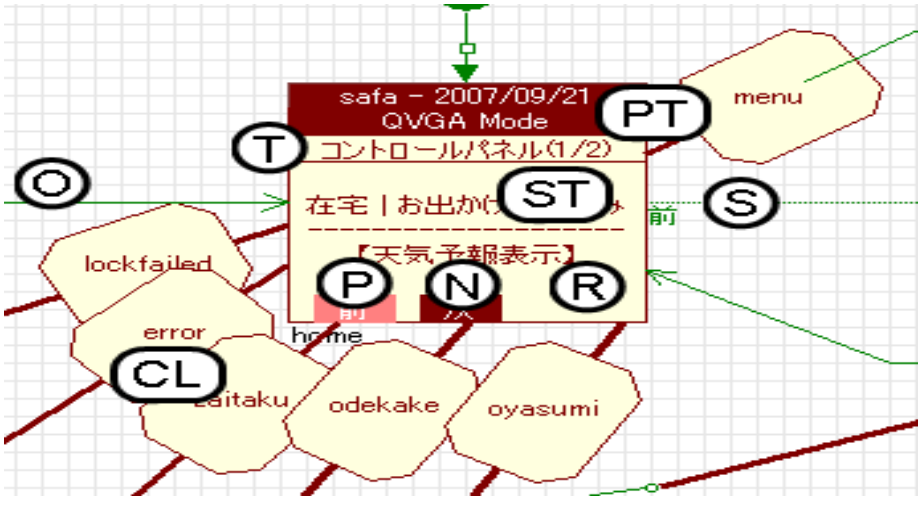

Fig. 12. エスケープ・セマンティックのインスタンス

複数の標準エレメントが、しばしば頻度の少ないパターンに再利用されることが判った。タイトル (T), 前のボタン(P), 次のボタン (N), 戻るボタン (R), 関連ページへのリンク (S), 他のページからのリンク (O) など。これらは、セミコントロールなモデリングを用いて、モックアップページの生成を可能にし、十分な情報をもって開発作業中の製品に統合される。

自由なモデリングは、シンプルなテキスト(ST), 条件によるリンク (CL), 新しいページタイプの定義/提案 (PT)で、提供される。 この例では、モデリング担当者は、２つの区分で、特別なフォントページを定義している。最初の区分では、3 つのボタンが含まれ、ホームオペレーションモードがワンタッチで変更できるようになり、2 つ目の区分では、天気予報を表示する為の仕掛けを含んでいる。

7.4 ツール化の制限

要求に対するカバレッジと、使用頻度のいずれかを基準に、ユニークな新機能に対して開発リソースを集中することなく、ツール化に際して、時間とコストに制限を設けた。このツール化の制限は、任意に設けられ、ドメインに対する知識や、現状のマーケットニーズに対する認識に応じて、定期的に再考された。

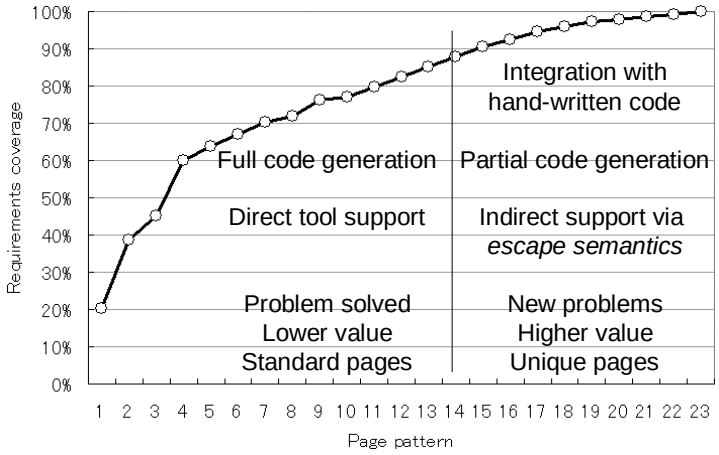

Fig. 13. ツール化の制限

7.5 実アプリケーションのサンプルモデル

Fig. 14 は、5 つの標準ページ・パターン (*menu*, *configuration*, *question*, *confirmation*, and *splash*) と、エスケープセマンティックのための未定義（ジョーカー）シンボルでモデル化された、Home Control Panel [5] の１ユーザインターフェイスである。全 45 ページ中、27 ページが５つの標準ページ・パターンでモデル化され、18 ページが未定義（ジョーカー）シンボルが必要であった。結果、ユーザインターフェイス・デザインツールは、60%の要求仕様をカバーしていることになる。要求を良く調べると、P1 と P2 の 2 つのパターンが明らかになり、これらをサポートすることで、7 ページを追加サポートできる。これは、34/45 = 75% の要求仕様をカバーすることになる。

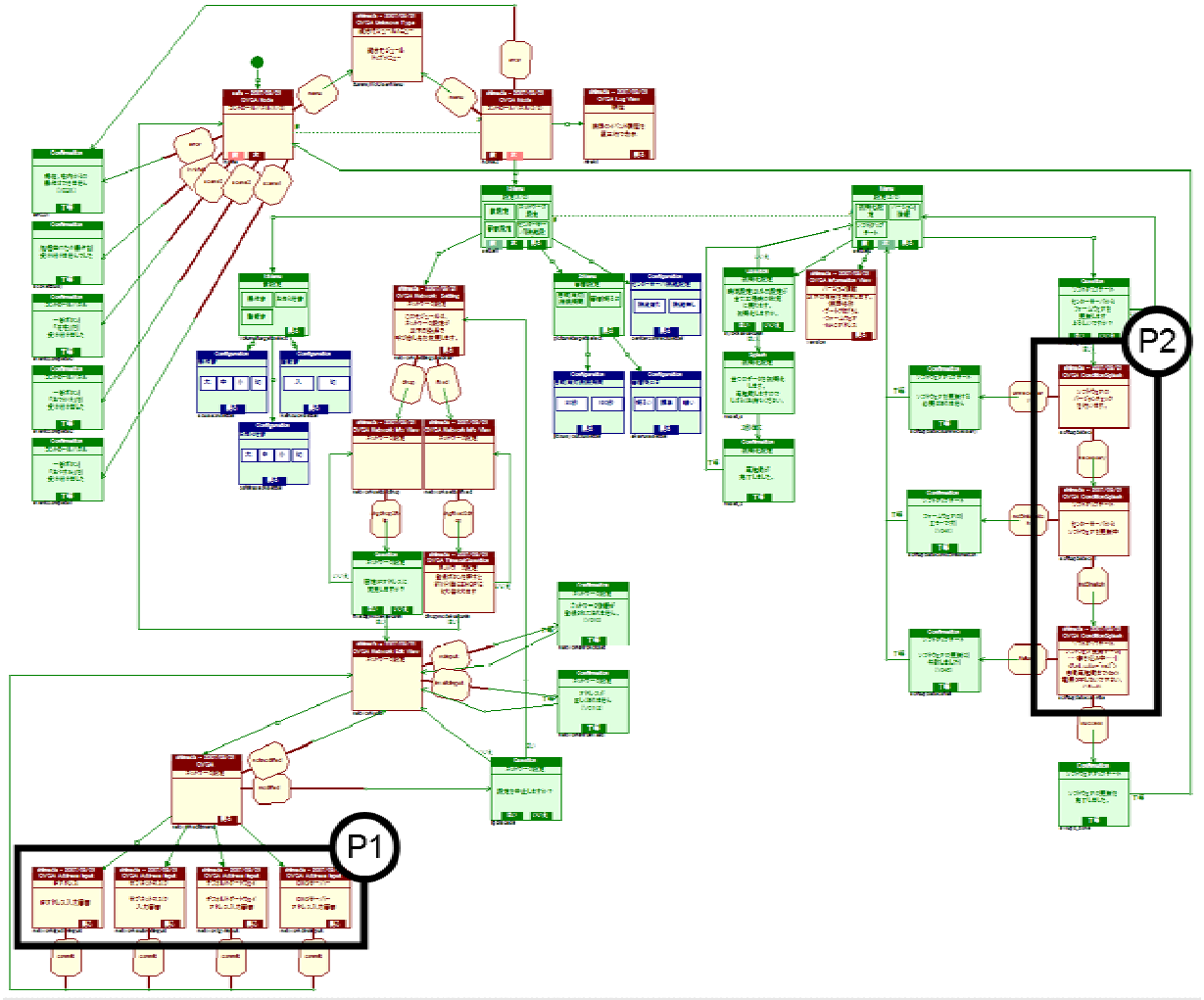

Fig. 14. 既存アプリケーションを５つのパターンと未定義（ジョーカー）シンボルでモデル化

6.保守性

技術的関心事に加え、廉価で進化させることができて、他者への技術移管が容易であるような、保守可能なツールの開発を求めていた。 大学生を6ヶ月のインターンシップとして迎える機会を利用し、いくつかの試みを実施した。彼のことを、フレッシュマンと呼ぶ。

8.1 簡素化

社内で 1 週間が過ぎ、フレッシュマンはもちろん製品について、内在するフレームワーク、プログラミング・ガイドライン、モデリングツールなど、何も知り得なかった。しかしながら、ツールに対する半日のトレーニングで、モデリングして、熟練者による実装コードと同じ品質の組み込みユ

ーザインターフェイスを、わずかな時間で構築できるようになった。　そしてこの試みは、特定のアプリケーションドメイン、プロセスエリアに於いて、標準クラスの従業員に熟練者のパワーを与える、我々のドメイン・スペシフィック・ツールの、能力の実証に役立つものであった。

8.2 順応性

フレッシュマンは、ツールとモデリング言語を理解できた。次に、我々のツールの順応性を評価するため、彼に対して、全く異なるターゲットプラットフォームに対するコード生成機能の実装を依頼した。

Supported platform　Embedded OS /CPU /MB RAM /QVGA /color /touch panel

Challenge platform　No OS /microcontroller /KB RAM /screen 101x64 pixels /monochrome /3-position knob (top, down, push)

MetaEdit+® の、GOPPRR メタモデリング・アプローチと、MERL スクリプト言語を、標準の説明書とサンプルのDSLを用いて3日で学習し、フレッシュマンは、ビルド/Flash書込み/実行、の自動化まで含めたCコード生成機能を、追加の3日で実装することが出来た。

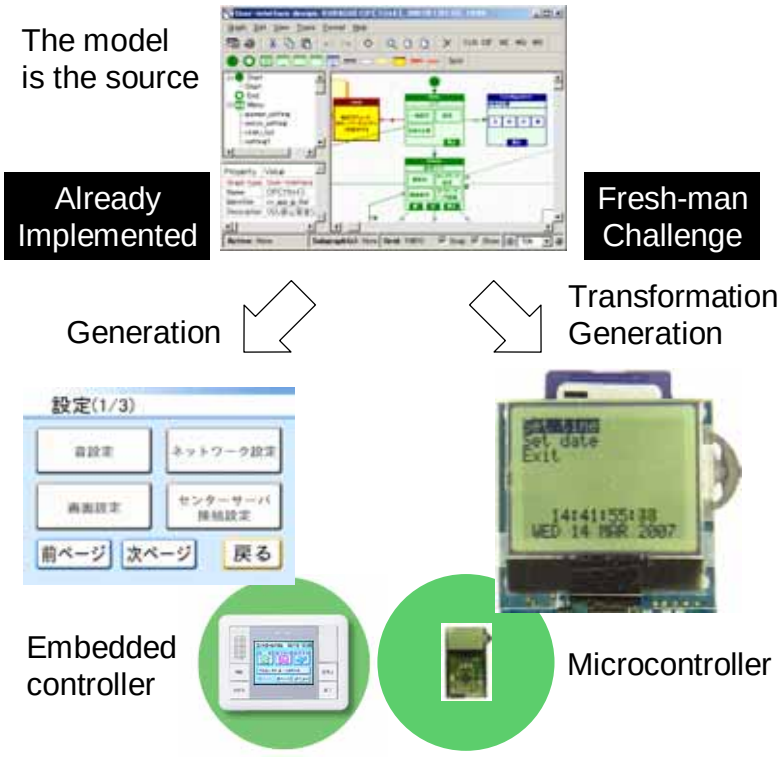

Fig. 15. ツールを新たなターゲットプラットフォームへ適応

8.3 他者への移管

3ヶ月を過ごしたフレッシュマンは、適切にツールのコア開発に参与できた。このことは、安定し、十分にドキュメント化された、GOPPRR と MERL の技術に対して、最小限のコーディングを選択した成果であると確信する。

まとめ

本資料では、従来型開発の 3〜5 倍の生産性向上を可能とする、ホームオートメーション市場向け日本語ユーザインターフェイスをモデル化し、生成する独自のユーザインタフェース・デザインツールを開発した事例を報告した。
そして、ツールの支持を円滑にするために重要な、いくつかの機能について紹介した：仮想－実在的モデリング、動的モデリング、オンサイトに於いてのツールの構成。更に、ツールの制約の概念と、エスケープ・セマンティックについて述べた。これらは、ツールにより提供される機能と、製品開発の実践で必要となることの、ギャップに対処するうえで役に立つ。最後に、ツールの保守性に対する課題に対して、フレッシュマンによる試みを実施し、調査した。このツールは、有効であり、雇用現場に於いて、一般的なスキルの担当者でも保守可能であることを実証した。
全体を通して、本資料で報告したユーザインタフェース・デザインツールの開発は、社内に於けるマイルストーンとなった。製品品質のソフトウエアを生成し、ソフトウエアの構築作業をビジュアルモデルからターゲットへ完全に自動化する。新たな製品開発への幕開けである。

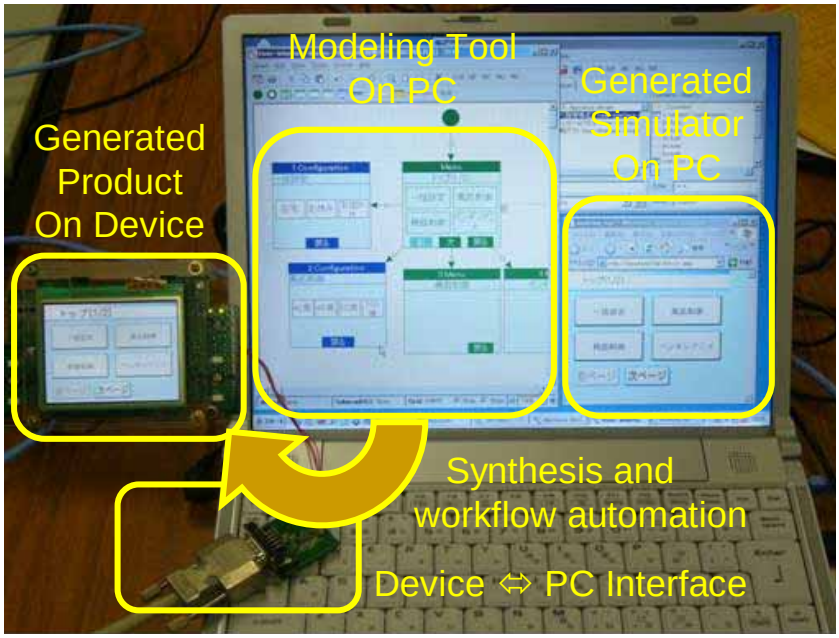

Fig. 14. First occurrence of complete synthesis and workflow automation for the embedded user-interfaces

8. References

1. Web news about MEW adding energy saving features to Lifinity®, http://plusd.itmedia.co.jp/lifestyle/articles/0705/15/news073.html
2. Kelly, S., Lyytinen, K., and Rossi, M.: MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE Environment. In: Proceedings of CAiSE'96, 8th Intl. Conference on Advanced Information Systems Engineering, Lecture Notes in Computer Science 1080, Springer-Verlag, pp. 1-21, 1996.
3. Kelly, S., and Pohjonen, R.: Interactive Television Applications using MetaEdit+. MDD-TIF07.
4. Safa, L.: The Practice of Deploying DSM - Report from a Japanese Appliance Maker Trenches. DSM'06
5. Lifinity® product home-page http://biz.national.jp/Ebox/kahs/index.html
6. MEW Corporate report: Lifinity® Home Network System, http://www.mew.co.jp/e/corp/ir/annual/2007repo/pdfs/05.pdf