

柔軟なソフトウェアの構築を可能にする
コンポーネントアーキテクチャ

筑波大学審査学位論文(博士)

1999

上田 哲郎

筑波大学大学院
経営・政策科学研究科 企業科学専攻

寄贈
上田哲郎氏

目次

序章	13
1 オブジェクト指向によるプログラミング	15
1.1 オブジェクト指向プログラミング	15
1.2 オブジェクト指向プログラミングの開発側面	17
1.2.1 “a-kind-of” 開発側面	17
1.2.2 “a-part-of” 開発側面	18
1.3 クラスライブラリを用いた開発	19
1.3.1 クラスライブラリとは	19
1.3.2 クラスライブラリの問題点	19
1.4 デザインパターンを用いた開発	20
1.4.1 デザインパターンとは	20
1.4.2 デザインパターンカタログ	20
1.4.3 デザインパターンの問題点	21
1.5 フレームワークを用いた開発	22
1.5.1 フレームワークとは	22
1.5.2 クラスライブラリとの違い	22
1.5.3 アプリケーションフレームワークの例	24
1.5.4 フレームワークを用いた開発の問題点	25
1.6 コンポーネントを用いた開発	26
1.6.1 コンポーネントを用いた開発とは	26
1.6.2 コンポーネントの例	27
1.6.3 コンポーネントを用いた開発の問題点	27
1.7 考察	29
2 柔軟な結合が可能なコンポーネント –Nuts	31
2.1 ホワイトボックス的プログラミングの特徴と問題点	31
2.2 ブラックボックス的プログラミングの特徴と問題点	34
2.3 ホワイトボックス的プログラミングが可能なコンポーネント	36
2.4 Nuts コンポーネントの結合	37
2.5 モジュール性に関する議論	38

2.6	サブクラス拡張の問題点とコンポーネント差分プログラミング	39
2.7	ブラックボックス的なプログラミングと比較した Nuts の拡張性	41
2.8	C++開発フレームワークとしての Nuts	43
2.9	考察	44
3	Nuts アーキテクチャ	45
3.1	Nuts の構造モデル	45
3.2	Nuts の制御モデル	49
3.2.1	メッセージによる制御	49
3.2.2	イベントによる制御	50
3.2.3	メッセージオブジェクト	52
3.2.4	メッセージの横取り	53
3.3	考察	56
4	Nuts のクラス階層	57
4.1	NutsCore クラス	57
4.1.1	Nuts コンポーネントの規約	57
4.1.2	NutsCore クラスのメンバ変数	58
4.1.3	NutsCore クラスの初期化フェーズ	58
4.1.4	NutsCore クラスでのメッセージ処理の流れ	62
4.1.5	NutsCore クラスで認識可能なメッセージ	63
4.1.6	メッセージの多重の横取り	64
4.2	NutsManager クラス	65
4.2.1	NutsManager クラスのメンバ変数	65
4.2.2	NutsManager クラスの初期化フェーズ	66
4.2.3	NutsManager クラスでのメッセージ処理の流れ	68
4.2.4	NutsManager クラスで認識可能なメッセージ	69
4.3	NutsApp クラス	69
4.3.1	NutsApp クラスのメンバ変数	69
4.3.2	NutsApp クラスの初期化フェーズ	70
4.3.3	NutsApp クラスで認識可能なメッセージ	71
4.4	考察	72
5	透明なコンポーネント	73
5.1	透明コンポーネントと親子関係の例外	73
5.2	メッセージの横取りと透明コンポーネント	74
5.3	NutsShadowManager クラス	75
5.3.1	構造モデルの例外	75
5.3.2	制御モデルの例外	76
5.4	NutsGroupManager クラス	78

5.4.1	NutsGroupManager クラスの初期化フェーズ	78
5.4.2	NutsGroupManager クラスを用いたメッセージ通信	79
5.5	考察	82
6	コンポーネント差分プログラミング	83
6.1	コンポーネント差分プログラミングとは	83
6.2	ベクターコンポーネント	86
6.3	NutsVector クラス	88
6.3.1	NutsVector クラスの初期化フェーズ	88
6.3.2	NutsVector クラスでのメッセージ処理	89
6.4	ベクターコンポーネントの例	90
6.4.1	ウインドウをスクロールするベクター	90
6.4.2	アイコンをドラッグする／投げるベクター	91
6.4.3	アイコンを捕獲するベクター	93
6.4.4	ベクターの挿入	93
6.5	ベクターを用いたアプリケーション開発	94
6.6	考察	94
7	Nuts によるアプリケーション開発	97
7.1	Nuts/Builder	97
7.1.1	Nuts/Builder とは	97
7.1.2	Nuts/Builder で導入した透明コンポーネント	97
7.1.2.1	workSheetManager/rootManager コンポーネント	97
7.1.2.2	dragSource/dropSite コンポーネント	102
7.2	DRMA 地図ドライバ	103
7.2.1	DRMA 地図ドライバとは	103
7.2.2	地図データの読み込み	103
7.2.2.1	DRMA 地図データベースの構成	103
7.2.2.2	メッシュデータのクラス	104
7.2.2.3	内部的なメッシュ番号とメッシュの相対位置計算	105
7.2.2.4	地図要素のクラス	106
7.2.3	画面表示とスクローリング	107
7.2.3.1	初期表示	107
7.2.3.2	バッファ内スクロール	107
7.2.3.3	バッファ外スクロール	108
7.2.3.4	データキャッシュ	108
7.2.4	全体構成	110
7.3	ニューラルネットワークシミュレータ	111
7.3.1	階層型ニューラルネットワーク	111
7.3.2	Nuts による階層型ニューラルネットワークシミュレータの開発	112

7.3.3	ニューラルネットワークシミュレータの構成例	114
7.4	ビジネスアプリケーション開発事例	115
7.5	考察	119
8	配送計画支援システム Navimos の開発	121
8.1	物流交通アプリケーションフレームワークの構想	121
8.2	フレームワーク Navimos とは	122
8.3	配送計画支援システム Navimos-Master の開発	123
8.3.1	最適な配送計画とは	123
8.3.2	最適化のアルゴリズム	124
8.3.2.1	配車最適化	124
8.3.2.2	配送コース最適化	125
8.3.3	データの準備	126
8.3.4	計画提示と修正	127
8.3.5	Navimos-Master の実装	128
8.3.6	Navimos-Master への動態把握機能の追加	131
8.3.6.1	動態把握機能の概要	131
8.3.6.2	動態把握機能の実装	131
8.4	Nuts による Navimos-Master の開発	134
8.4.1	Nuts 版 Navimos-Master の構成	134
8.4.2	Nuts 版 Navimos-Master の実装	139
8.4.3	Nuts 版 Navimos-Master の拡張	139
8.5	考察	140
9	議論	143
9.1	Nuts によるソフトウェア開発効率の向上に関する議論	143
9.1.1	アイコン投げシェルの開発における Nuts の効果	143
9.1.2	Navimos の開発における Nuts の効果	149
9.2	デザインパターンとの比較	149
9.2.1	修飾に関するパターンとの比較	152
9.2.2	機能の変更に関するパターンとの比較	155
9.2.3	コンポーネントをグループ化するパターンとの比較	157
9.2.4	コンポーネント群を単一のコンポーネントとして扱うパターンとの比較	157
9.3	他のフレームワークとの比較	158
9.3.1	フレームワーク ET++ との比較	158
9.3.2	IntelligentPad との比較	160
9.4	コンパイル時 MOP を用いた拡張との比較	160

目次	5
10 結論	161
10.1 Nuts フレームワークの今後の課題	161
10.1.1 ベクターを修飾するベクター	161
10.1.2 コンポーネント内部の拡張	162
10.1.3 適用範囲の拡大	162
10.1.4 分散化	165
10.2 結論	165
謝辞	167
業績リスト	176
A Nuts の全クラス	179

目次

1.1	オブジェクトの開発側面	18
1.2	クラスライブラリ (a) とフレームワーク (b) の位置付け	23
1.3	フレームワークの利用	24
1.4	オブジェクトグラフ	29
2.1	オブジェクト a,b の参照関係	32
2.2	ActiveX コンポーネントの取得	35
2.3	オブジェクトの結合	37
2.4	ベクターコンポーネントによる拡張	40
2.5	ActiveX コンポーネントの使用と拡張	42
3.1	Nuts の木構造	46
3.2	コンポーネントの親子関係	47
3.3	構造の記述例	47
3.4	nutsCreateMotifApp の出力例	48
3.5	自動生成されたウインドウ	48
3.6	メッセージ探索	51
3.7	イベントモデル	52
3.8	リスナオブジェクトのメンバ	52
3.9	メッセージオブジェクトのメンバ	53
3.10	メッセージ送受信者間でのやりとり	53
3.11	メッセージの横取りによるプリンタの切替え	54
3.12	メッセージ横取りの方法	55
3.13	メッセージの横取り	55
4.1	Nuts の基底クラス群	58
4.2	nutsCreateNewClass によって自動生成されたヘッダファイル	59
4.3	nutsCreateNewClass によって自動生成されたソースファイル	60
4.4	NutsCore クラスの初期化フェーズ	63
4.5	メッセージ授受の流れ	64
4.6	メッセージの多重の横取り	66
4.7	NutsManager クラスの初期化フェーズ	67

4.8	親子の参照関係	68
4.9	NutsApp クラスの初期化フェーズ	70
5.1	透明コンポーネントの親子関係	74
5.2	透明コンポーネントからの横取り	75
5.3	透明コンポーネントの参照関係	77
5.4	グループ構成	79
5.5	キャンバスウインドウ	80
5.6	キャンバスウインドウの構成	81
5.7	部品化したキャンバスウインドウ	81
6.1	オブジェクト指向によらない拡張	84
6.2	従来のオブジェクト指向による拡張	84
6.3	ベクターによる拡張	84
6.4	コンポーネント差分プログラミングの拡張手法	86
6.5	被修飾コンポーネントの探索	87
6.6	ベクターによる多重修飾	89
6.7	透明コンポーネントのクラス階層	90
6.8	スクロール機能の追加と多重化	91
6.9	NutsHorizontalScroller	92
6.10	アイコンを投げる場合の制御	93
6.11	GUI シェル	95
6.12	ベクターコンポーネントによる GUI シェルの実装	95
7.1	Nuts/Builder の部品積み上げの様子	98
7.2	Nuts/Builder の画面イメージ	99
7.3	workSheetManager と rootManager	101
7.4	dragSource と dropSite	102
7.5	DRMA 地図ビューアー	104
7.6	mesh コードの規則	105
7.7	DRMA 地図データベースのメッシュ構造	106
7.8	mesh クラスの階層構造	107
7.9	地図要素クラスの階層構造	107
7.10	オブジェクト間の参照関係	108
7.11	スコープと画像バッファの関係	109
7.12	データのスワップ	109
7.13	プログラムの全体構成	110
7.14	階層型ニューラルネットワークの構成	111
7.15	Nuts における階層型ニューラルネットワークのモデル	112
7.16	Nuts における階層型ニューラルネットワークの部品構成	112
7.17	4bits → 2bits エンコーダの部品構成	116

7.18	4bits → 2bits エンコーダの操作パネル	116
7.19	4bits → 2bits エンコーダの構造記述ファイル	117
7.20	2bits → 4bits デコーダの構造記述ファイル	118
7.21	Nuts ライブラリを用いたコード比率	119
7.22	ビジネスアプリケーションの一つの画面イメージ (秘匿義務のため画面の一部をぼかしている)	120
8.1	Navimos のサポート分野	123
8.2	セービング値	125
8.3	Navimos の時間制約付きセービング法	126
8.4	巡回セールスマン問題のメタ解法	127
8.5	Navimos-Master の操作パネル	128
8.6	nodeRec クラスのメンバー	129
8.7	truckRec クラスのメンバー	129
8.8	トラック属性の変更ダイアログ	130
8.9	動態把握機能をもった Navimos-Master	132
8.10	動態把握システムの構成	133
8.11	携帯電話を使った動態把握システムの構成	133
8.12	パケット通信を使った動態把握システムの構成	133
8.13	Nuts 版 Navimos の部品構成	136
8.14	複製した後の Nuts 版 Navimos の部品構成	136
8.15	Nuts 版 Navimos の構造記述ファイル	137
8.16	Nuts 版 Navimos の画面イメージ	138
8.17	ベクター挿入のためのコーディング	139
9.1	アイコン投げシェルのイメージ	144
9.2	Nuts を用いたアイコン投げシェルの構造記述ファイル	146
9.3	Nuts を用いたアイコン投げシェルの構造	146
9.4	Nuts を用いない場合の構造	147
9.5	Decorator パターンの使用	152
9.6	Decorator パターンの使用	154
9.7	Vector コンポーネントのインタフェース	154
9.8	被せると横取るの相違	155
9.9	Strategy パターンの実装	156
9.10	ET++を用いた Hello world	159
9.11	Nuts を用いた Hello world	159
10.1	OpenC++のメタクラスの定義	163
10.2	OpenC++のメタクラスの使用	163
10.3	Java 版 Nuts のコーディング例と実行結果	164

表 目 次

1.1	デザインパターンカタログの内容 ([24] より引用)	21
1.2	主なアプリケーションフレームワーク	25
1.3	主なコンポーネント製品	28
2.1	各コンポーネントの結合度	39
3.1	メッセージ探索キー	50
4.1	Nuts コンポーネントの規約	61
4.2	NutsCore のメンバ変数	62
4.3	NutsCore クラスで認識するメッセージ	65
4.4	NutsManager のメンバ変数	66
4.5	NutsManager クラスで認識するメッセージ	69
4.6	NutsApp のメンバ変数	69
4.7	ルートコンポーネントのレパートリ	71
4.8	NutsApp クラスで認識するメッセージ	71
4.9	Nuts ライブラリの内訳	72
7.1	Nuts/Builder で用いた部品のクラス (数字は行数)	100
7.2	ニューラルマネージャの受理するメッセージ	113
7.3	Nuts のニューロン部品	114
7.4	4bits → 2 ビットエンコーダの学習パターン	114
7.5	4bits → 2 ビットエンコーダのテスト結果	114
7.6	Nuts を用いた開発例	115
9.1	アイコン投げシェルの開発におけるデータ比較	145
9.2	アイコン投げシェルのクラス (数字は、コンパイル時に知っている必要がある他クラス定義の数)	145
9.3	アイコン投げシェルの開発における比較	148
9.4	Navimos の開発におけるデータ比較	150
9.5	Navimos 開発における比較	150
9.6	Navimos の開発に用いた部品リスト	151

10.1 Java 版 Nuts のコンポーネント群	165
--------------------------------------	-----