M-020

カレンダ情報を操作可能なローコードシステムの提案
A Low-code System for Manipulation of Calendar Information

木村 聡志[1)]　乃村 能成[2)]
Satoshi Kimura　Yoshinari Nomura

1. はじめに

Google カレンダー [1] をはじめとするカレンダシステムは，業務やプライベートの場面にかかわらずよく利用されるが，データの操作に向いていないという問題がある．これは，既存のカレンダシステムは予定を再利用可能なデータとして操作するツールとしては想定されていないためである．カレンダを集計したり特定の予定を一括でコピーしたりといった複雑な操作をするためには，カレンダ情報をデータベースとしてプログラミングを行う必要がある．現在，カレンダ情報をデータベースとしたプログラミングは，Google カレンダー API を用いることで実現できる [2]．しかし，プログラミング経験のないユーザにとっては Google カレンダー API を用いたプログラミングは困難である．

そこで本稿では，プログラミング経験のないユーザでも利用できるカレンダ情報を操作可能なローコードシステムを提案する，

2. カレンダ情報を操作可能なローコードシステム

提案システムはカレンダ情報の操作に特化したプログラミングを可能とする．また，テキストによるプログラミングでは，プログラミング経験のないユーザは利用が困難であるため，GUI 操作によってプログラミングを実現する．提案システムを実現するためには，バックエンドとなる問合せ言語が必要となるため，ドメイン特化言語 (DSL) の設計が必要である．また，GUI 操作によるプログラミングのためのユーザインタフェースを設計する必要がある．

3. プログラミングに用いる DSL

3.1 DSL への要求

本章では DSL を設計する上での方針を示す．まず，提案システムを用いてプログラミングを行うユースケースを考える．

まず，自分の予定を一部書き換えて他者に共有する操作を考える．たとえば，「マイカレンダ」にある予定のうち，『買い物』という予定の予定名を『私用』に置換し，すべての予定を「同僚のカレンダ」に公開する．この操作は次の手続きになる．

(1) カレンダ「マイカレンダ」の全予定をデータベースから取得
(2) 取得した各予定に以下の処理を行う
(2-A) 予定名が『買い物』のとき，その予定名を『私用』に書き換える
(2-B) カレンダ「同僚のカレンダ」に予定を追加

1） 岡山大学大学院自然科学研究科，Graduate School of Natural Science and Technology, Okayama University
2） 岡山大学学術研究院自然科学学域，Faculty of Natural Science and Technology, Okayama University

DSL には，(1)，(2-B) のようにカレンダの予定の取得やカレンダへの予定の追加といったデータベースへの問合せ機能が必要となる．また，(2-A) のように予定を加工する機能が必要となる．

次に，カレンダの予定を集計する操作を考える．たとえば，カレンダを集計し先月行われた会議の合計時間を算出する場合を考える．この操作は次の手続きになる．

(1) カレンダ「マイカレンダ」の全予定をデータベースから取得
(2) 取得した予定から「先月」および「予定名に『会議』を含む」という条件で予定を抽出
(3) 抽出した全予定の合計時間を算出

例のように，予定名に特定の言葉を含む予定のみに対して操作を行うには，(2) のように予定の集合から指定した条件の予定を抽出する機能が必要である．また，(2) では「先月」という条件で絞り込んでいるように，カレンダを扱う上では月や週，曜日といった暦に関する要素は重要となる．このため，暦に関する要素はリテラルとして表現できることが求められる．

予定の加工を行う際は，例に示したように，まずカレンダから予定を取得し，そして取得した予定や絞り込んだ予定集合に対して加工を行う．つまり加工操作は複数の予定に対する一括操作が前提となる．予定の集合に加工操作を行う際，処理内部では一つ一つの予定に対する操作を繰り返し行う．しかし，利用者にとって集合に対する操作を繰り返しを用いて記述することは直感的ではない．そこで，予定集合に対して一括操作を行う機能が求められる．

DSL への要求をまとめると以下の通りである．

(要求 1) データベースへの問合せ機能
(要求 2) 予定の絞り込み機能
(要求 3) 予定に対する加工機能
(要求 4) 予定集合への一括操作を行う機能
(要求 5) 暦に関する要素を表現するリテラル

3.2 DSL の記述例

3.1 節の要求を満たす DSL の記述例を示す．**表 1** に**リスト 1** と**リスト 2** で使われている DSL の記法を示す．表 1 の通番 1,2 は (要求 1) のデータベースへの問合せ機能である．また，通番 3 は (要求 2) の絞り込み機能，通番 5 は (要求 3) の予定加工機能を提供する．

リスト 1　共有操作のプログラム例

```
1  get_events("マイカレンダ").map(function(e) {
2    if (e.summary == "買い物") {
3      e.replace(summary, "私用")
4    }
5    insert_events(e, "同僚のカレンダ")
6  })
```

表 1　リスト 1 とリスト 2 で使われている DSL の記法

通番	DSL	説明
1	get_events	指定したカレンダから全予定を取得
2	insert_events	指定したカレンダに予定を追加
3	filter	指定した条件で予定を絞り込む
4	map	予定集合に対する一括操作
5	replace	予定のプロパティの値を変更
6	M”lastmonth”	先月を意味するリテラル
7	summary	予定名を表す予定のプロパティ
8	total_hours	予定の合計時間を算出
9	if	条件分岐
10	match	指定した文字列を含むかの判定
11	print	値を表示

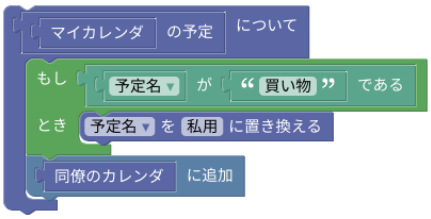

図 1　map 操作のブロックを使ったプログラム

リスト 2　集計操作のプログラム例

```
print(get_events("マイカレンダ")
.filter(M"lastmonth" & summary.match("会議"))
.total_hours)
```

リスト 1 とリスト 2 は 3.1 節で挙げた共有操作と集計操作を DSL で記述した例である．リスト 1 の 2 行目から 5 行目までの一連の処理は，一つの予定に対しての操作を表している．表 1 の通番 4 はこの一つの予定に対する操作を予定集合全体に適用させることを意味しており，(要求 4) の一括操作を表現している．また，通番 6 は先月を意味するリテラルであり，(要求 5) を満たす．リスト 2 に示すように，通番 6 を用いることで先月の予定を絞り込むという記述を簡潔に表現できる．

4. プログラミングのユーザインタフェース

4.1 ユーザインタフェースの設計

3 章で述べた DSL を記述するためのユーザインタフェースを設計する．ユーザインタフェースはプログラミング経験のないユーザでも利用できる必要がある．そこでビジュアルプログラミング構築のためのライブラリである Blockly[3] を用いてユーザインタフェースを設計する．Blockly は連結するビジュアルブロックを使用し，任意のテキストプログラミング言語のコードを生成できる．視覚的なブロックを連結させて手続き的な処理を記述できるため，プログラミング経験のないユーザでも利用できるという要求を満たす．

図 2　先月の会議の合計時間を求めるプログラム

表 2　図 1 に用いられているブロックと生成する DSL

通番	ブロック	DSL
1	Events について / Statement	*Events*.map(function(e){ *Statement* })
2	Calendar の予定	get_events(*Calendar*)
3	もし Expression / とき Statement	if(*Expression*){*Statement*}
4	Expression が Expression である	*Expression* == *Expression*
5	Property を □ に置き換える	e.replace(*Property*, *String*)
6	Calendar に追加	insert_events(*Calendar*, e)

表 3　図 2 に用いられているブロックと生成する DSL

通番	ブロック	DSL
1	Calendar の予定	get_events(*Calendar*)
2	Events を Expression で絞り込む	*Events*.filter(*Expression*)
3	Property に String が含まれる	*Property*.match(*String*)
4	Events の合計時間	total_hours(*Events*)
5	Expression を表示	print(*Expression*)

4.2 ビジュアルブロックの設計

4.1 節で述べたように，ユーザインタフェースはプログラミング経験のないユーザでも利用できる必要がある．このようなユーザにとっては変数の扱いは難しく，直感的に記述できるためには変数の使用は避けるべきである．そこで**表 2** の通番 1 に示す map 操作のブロックを考える．**図 1** にリスト 1 をビジュアルブロックを用いて記述したプログラム例を示す．図 1 に示すように，このブロックを用いることで，「マイカレンダ」から取得したすべての予定を変数を介さずに値を加工し，「同僚のカレンダ」へ追加する処理を記述している．このとき DSL の文法上では map 関数内で変数 e が定義されている．変数 e は表 2 の通番 5 と通番 6 の DSL で暗黙的に使われており，ブロック上では省略されている．

図 2 にリスト 2 をビジュアルブロックを用いて記述したプログラム例を示す．**表 3** の通番 4 は予定集合の合計時間を求めるブロックである．このブロックはユーザに繰り返しの処理を意識させずに，予定集合に対する一括操作を表現している．

5. おわりに

本稿では，カレンダ情報を操作可能なローコードシステムを提案した．まず，提案システムの概要について述べた．その後，プログラミングに用いる DSL を設計した．最後に，プログラミングのユーザインタフェースを設計した．

残された課題として，提案システムの有用性の評価がある．

参考文献

[1] Google Inc.: Google Calendar, Google (online), available from ⟨https://www.google.com/calendar/⟩ (accessed 2022-06-23).

[2] Darby, A.: Using Google Calendar to Manage Library Website Hours, *Code4Lib Journal*, No. 2 (2008).

[3] Google Inc.: Blockly, Google (online), available from ⟨https://developers.google.com/blockly⟩ (accessed 2022-06-23).