# オブジェクト指向言語の歴史


えムナウ （児玉宏之）

http://mnow.jp/

http://mnow.wankuma.com/

http://blogs.wankuma.com/mnow/

http://www.ailight.jp/blog/mnow/

.NET UX Lab
.Net ユーザーエクスペリエンス研究所

えムナウのC#
プログラミングのページ

わんくま同盟

# アジェンダ

- はじめに
- 時代的考察
- 神は死んだ
- ダイクストラはかく語りき
- 悦ばしき知恵
- 曙光（しょこう）
- まとめ

# はじめに

- オブ熱も3回目となり今までいろいろなアプローチで発表を行ってきている。
- しかし、オブジェクト指向の現在の状態を説明することがほとんどで、歴史的な側面で解説しどういう事情でこういう形式になっていったかを説明してこなかった。
- 今回は歴史を踏まえ将来の発展性を考えてみたいと思う

# 時代的考察

- コンピュータ言語
  - 1954年 FORTRAN　科学技術計算
  - 1958年 ALGOL　　研究目的
  - 1960年 COBOL　　事務処理
  - 1960年 LISP　　関数型言語
  - 1964年 BASIC　　初心者向け
  - 1966年 BCPL　　1972年C言語へ発展
  - 1966年 PL/I 科学技術事務処理共用

# 時代的考察

- FORTRAN　科学技術計算　1954年
  - ほぼ数学の数式通りに計算式を記述できる。
  - 科学技術計算に向いた逐次型の手続き型言語。
  - 暗黙の型宣言でIからNで始まる変数は整数型、ループカウンタにｉｊｋｌｍｎを使うのはこの頃の習慣による。
  - スーパーコンピュータでのプログラミング言語としてよく用いられた、その為自動的にベクトル命令にする機能追加されている。

# 時代的考察

- ALGOL　研究目的　1958年
  - 抽象的なアルゴリズムを手続きとして記述できる逐次型の手続き型言語。
  - 初めて再帰呼び出しが可能なプログラミング言語。
  - 値渡しとALGOLに特徴的な名前渡しがある、名前渡しは遅延評価の一種。
  - begin end の入れ子によるブロック構造化のルーツである。

# 時代的考察

- COBOL 事務処理　1960年
  - アメリカ国防総省によって、事務処理用の共通言語の開発が提案された官制言語。
  - 自然言語（英語）に類似した文章。
  - 見出し部やコンピュータ環境部などのプログラムから見ると注釈文の要素の強い部分も含んでいるが、データ部や手続き部のある、逐次型の手続き型言語。
  - 階層構造を持つレコードのデータ構造の定義が可能

# 時代的考察

- LISP　　関数型言語　1960年
  - 二分木のツリー構造からなる関数型言語。
  - ラムダ式（1930年代に考案された）の計算モデルを紙の上で表現するための記法として考案、1960年にコンピュータ上で実装される。
  - 計算可能な関数が表現でき正しく評価されチューリングマシンと等価な数理モデルとされているが、二分木で表すことの複雑さ（カリー化）やカッコの多さ、不動点演算子（Yコンバイン）など難解なことも多い。

# 神は死んだ

- ソフトウェア危機（software crisis）
  - 1960年代
    - 計算機の性能向上
    - 大規模なソフトウェア
    - コードの複雑化
    - ソフトウェア開発コストが上昇
    - ソフトウェア開発工程の体系化
    - **ソフトウェア工学の誕生**

# 神は死んだ

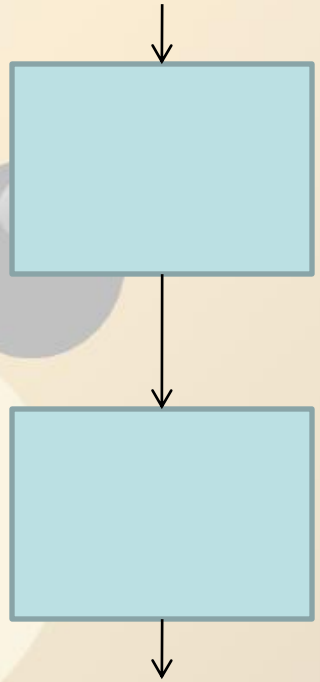
- ソフトウェアの再利用、部品化
  - 構造化プログラミング
  - 手続きや関数
  - 構成単位のブラックボックス化
  - 部品化を推進する仕組み
  - 弱い結合度、強い凝集度
  - ロジックの整理と再分化
  - 基本データ型からは脱却できず

# ダイクストラはかく語りき

- **構造化プログラミング**
  - メイン処理は大まかに記述
  - サブルーチンで詳細を実装
    - 同じ値を渡せば常に同じ結果が得られる
    - データと機能は直交（データを機能で順に処理加工していく）
    - プログラムのモジュール性を高める

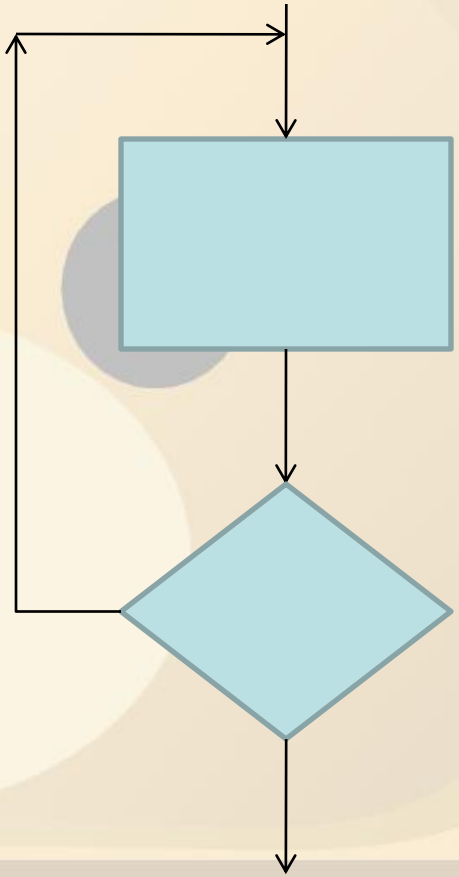
# ダイクストラはかく語りき

- **構造化プログラミング構成要素**
  - 順次
    - プログラムに記された順に逐次処理を行なっていく



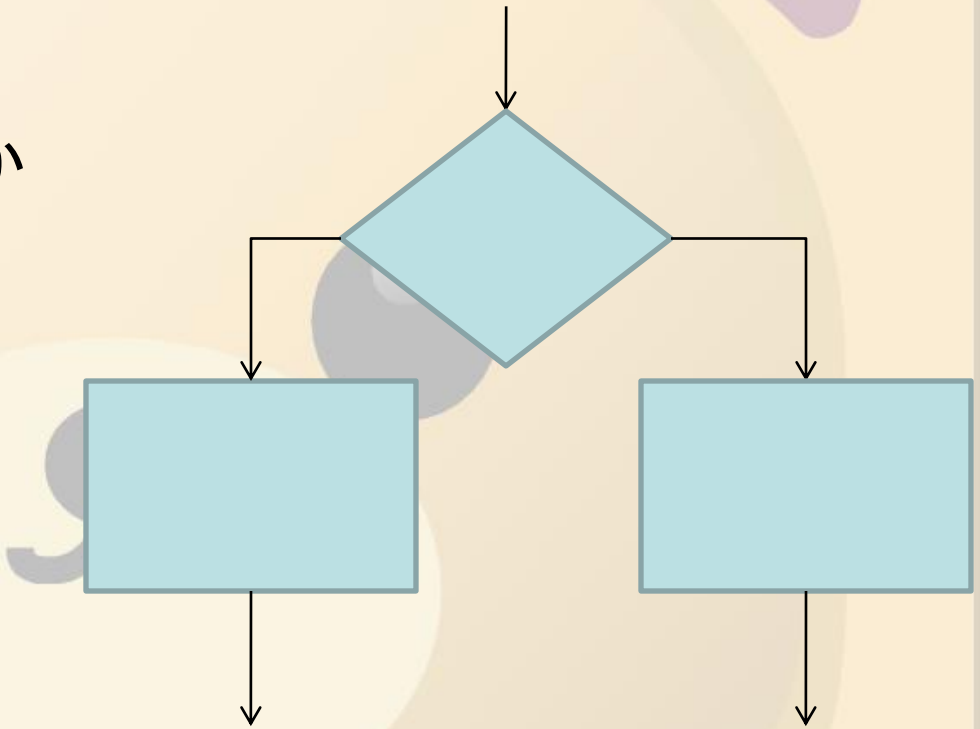
# ダイクストラはかく語りき

- **構造化プログラミング構成要素**
  - 反復
    - 一定の条件が
      満たされている間
      処理を繰り返す



# ダイクストラはかく語りき

- **構造化プログラミング構成要素**
  - 分岐
    - 条件が成立するか
      どうかで
      処理を振り分ける

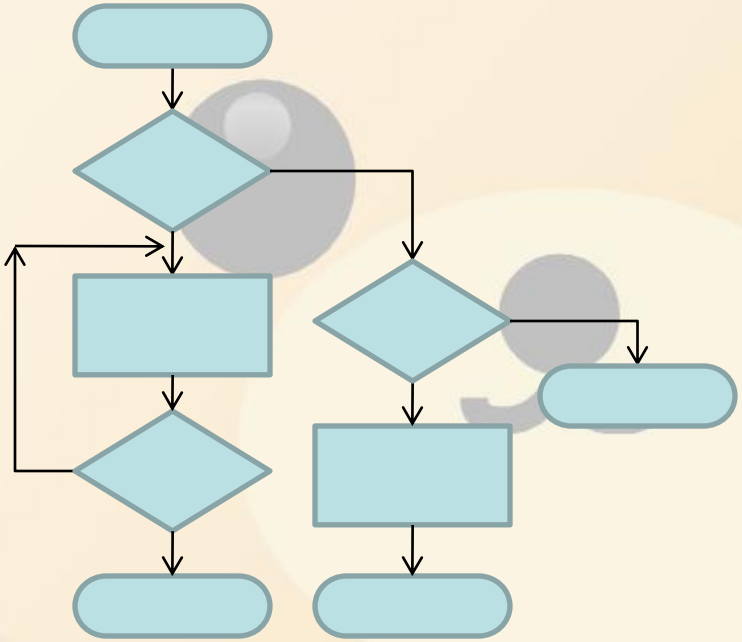


# ダイクストラはかく語りき

- **Go to 文有害論**
  - プログラムの処理の流れを変え、プログラムリスト上の特定の場所へ移行させる。
  - プログラムを記述する上で非常に強力だが、プログラムの構造がつかみにくい難点がある。
  - 構造化プログラミングでは基本構造を上手に使えばgoto文自体なくてもアルゴリズムを記述できる。

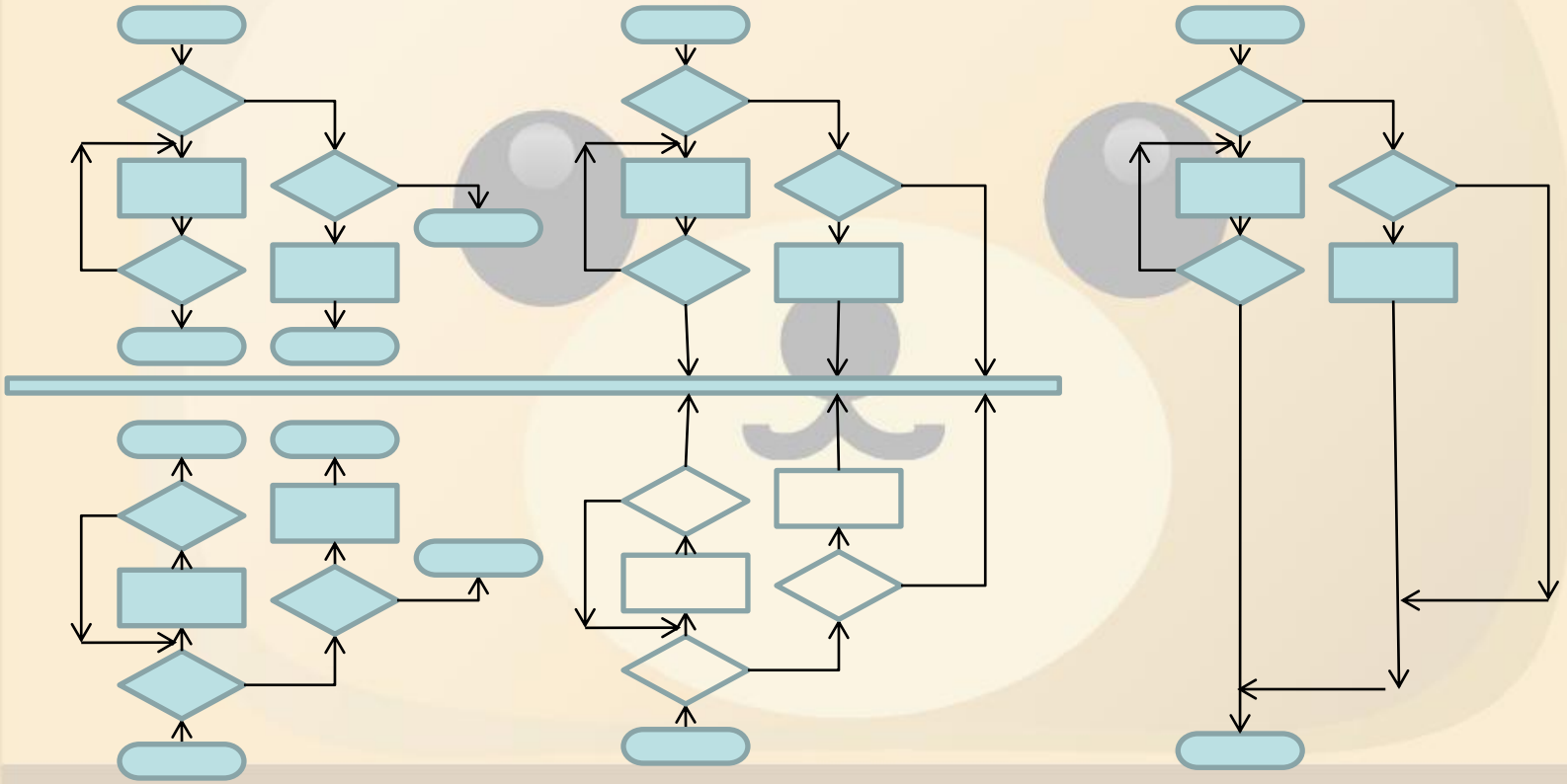
# ダイクストラはかく語りき

- **構造化の鏡**
  - たとえばこんな構造のプログラム



# ダイクストラはかく語りき

- **構造化の鏡**
  - 基本構造で一か所の出口アルゴリズムを記述



# 悦ばしき知恵

- コンピュータ言語
  - 1967年 Simula67　シミュレーション
  - 1972年 Smalltalk　AltoのOS
  - 1978年 Modula-2　モジュールの概念
  - 1979年 C++　C言語の拡張
  - 1990年 Python　スクリプト言語
  - 1991年 Java　仮想マシン上で動作
  - 1993年 Ruby　スクリプト言語
  - 2000年 C#　仮想マシン上で動作

# 悦ばしき知恵

- Simula67　　シミュレーション　1967年
  - ALGOL 60 を拡張したシミュレーション用言語。
  - クラスとオブジェクトという概念をもつが、シミュレーションを行う単位という意味合いとシミュレーション前後の状態を保持するための目的で、オブジェクト指向プログラミングのスタイルという意味合いではない。

# 悦ばしき知恵

- Smalltalk　　1972年
  - すべてをオブジェクトとそれらの間で交わされるメッセージ送信によって表現することを中心とするOSの概念も包括する環境。
  - オブジェクト指向という概念を構築した最初の言語。
  - このOSの環境は Mac や Windows というその後のパソコン環境に多大な影響を与えた、そういう意味では Mac や Windows という OS もオブジェクト指向である。

# 悦ばしき知恵

- Modula-2　　1978年
  - 汎用手続き型言語、構文の多くはPascal言語に基いたもの、モジュールの概念を追加し、分割コンパイルやソフトウェア部品のライブラリ化による再利用が可能。
  - 並行処理を可能にするコルーチンや、データ抽象化の機能を持つ。

# 悦ばしき知恵

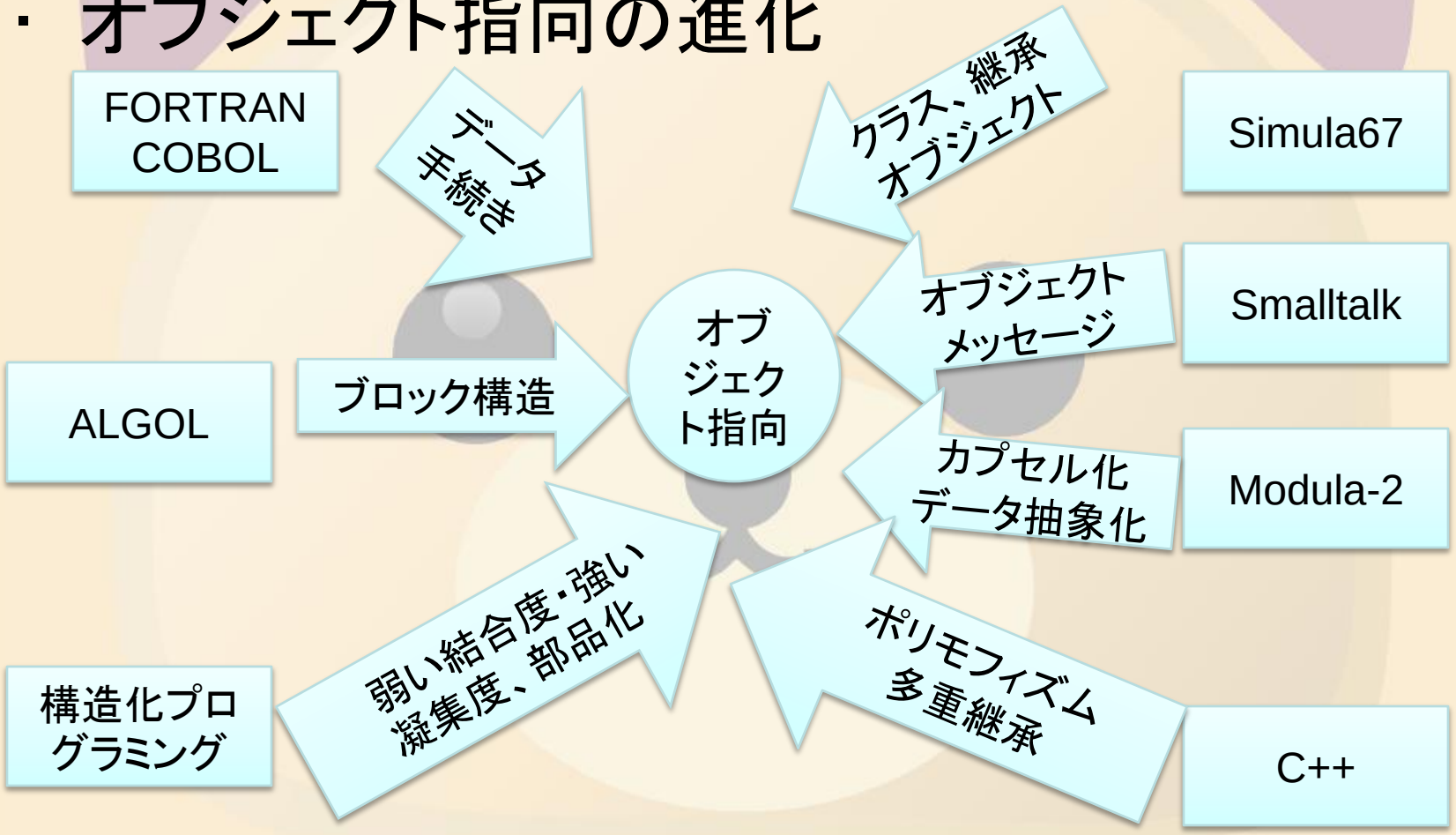
- C++　　1979年
  - C言語の拡張として開発されたマルチパラダイムプログラミング言語。
  - クラス、インターフェース、仮想関数、多重定義、多重継承、テンプレート、例外処理というオブジェクト指向言語としての一通りの機能を持つ。
  - 手続き型プログラミング・データ抽象・オブジェクト指向プログラミングばかりかジェネリックプログラミング・関数型プログラミング・メタプログラミングとしての拡張を模索。

# 悦ばしき知恵

- C＃　　2000年
  - .NET戦略の一環として開発されたマルチパラダイムプログラミング言語。
  - 手続き型プログラミング・データ抽象・オブジェクト指向プログラミングのみならず、ジェネリックプログラミング・関数型プログラミング・宣言型プログラミング・ドメイン特化モデリングによるコーディング補助の要素も取り入れていっている、管理されたアスペクト指向プログラム・並列処理の要素も模索されている。

# 曙光

- オブジェクト指向の進化



# 曙光

- 今後の発展の方向
  - マルチパラダイム
  - 非依存処理
  - 自動プログラミング
  - 分散コンピューティング
  - 並列コンピューティング
  - ユビキタス
  - ケースツール

# 曙光

- マルチパラダイム　言語構造
  - 手続き型プログラミング
  - 宣言型プログラミング
    - HTML・XML・SQL文も含める
    - 関数型プログラミング
      - 純粋関数型言語は変数を持たない
    - 論理プログラミング
      - 結論となるゴールを与え達成可能な解を求める
    - 制約プログラミング
      - 制約条件を与えそれを満たす解を求める

# 曙光

- マルチパラダイム　動作形態
  - 逐次実行型プログラミング
    - 起動後、順次命令を実行し、終了する
  - イベント駆動型プログラミング
    - イベントの発生により命令を実行する
  - データフローアーキテクチャ
    - データの変化により命令を実行する
  - トークン駆動型アーキテクチャ
    - プログラム間・CPU間のトークンの受け渡しにより命令を実行する

# 曙光

- マルチパラダイム
  モデリングと自動コーディング
  - ドメイン特化モデリング
    - ツールボックスに専用パーツを置き、プロパティを変更し線でつなぐことで関係を記述する
  - インテンショナルプログラミング
    - ツールボックスに専用パーツを置き、動作を記述する
    - 意図を図にすることに重きを置く

# 曙光

- マルチパラダイム　横断性
  - アスペクト指向プログラミング
    - クラス間を横断する機能を個々のクラス外に記述する
  - ジェネリックプログラミング
    - データ形式に依存しない処理を記述する

# 曙光

- マルチパラダイム　並列化
  - 分散コンピューティング
    - 複雑な計算などをネットワークを介して複数のコンピュータを利用して行う
  - 並列コンピューティング
    - 一つのコンピュータ上の複数プロセッサ協調
  - グリッド・コンピューティング
    - コンピュータ資源を結びつけひとつと見せる
  - ユビキタスコンピューティング
    - あらゆる場所であらゆるモノにコンピュータが組み込まれ、コンピュータ同士が協調動作する

# まとめ

- コンピュータはフォン・ノイマン型と言われる逐次実行型のシステムである
- プログラム言語はコンピュータにとって自然な手続き型として発展し速度や規模の増大によりソフトウェア危機と言われる混沌とした時代を迎える。
- ソフトウェア工学が議論され構造化プログラミングにより細分化や部品化・スパゲッティからの解放を果たした。

# まとめ

- オブジェクト指向の要素が色々と考えだされC++で集大成されその後の汎用言語はそれを引き継いで発展させていった。
- それでは、今はソフトウェア危機はおさまっているのか？
- オブジェクト指向に不足しているところをマルチパラダイムの時代は今後も発展させていくことであろう。