# 環境要素間の依存関係に基づく
# 部品仕様の構造化法

北陸先端科学技術大学院大学
海谷 治彦　　落水 浩一郎

May 21, 1998

# 発表の概要

- 背景と目的 ～ 部品の形式仕様が必要性，環境を考慮する必要性.
- 環境を考慮した部品仕様が有効である例.
- 上記の記述の問題点
  – 記述の複雑さ，汎用性の欠如.
- 環境を考慮した部品仕様の一般化
  – 部品の内容に注目して (SecurityManager の例).
- まとめと今後の課題.

# 背景と目的

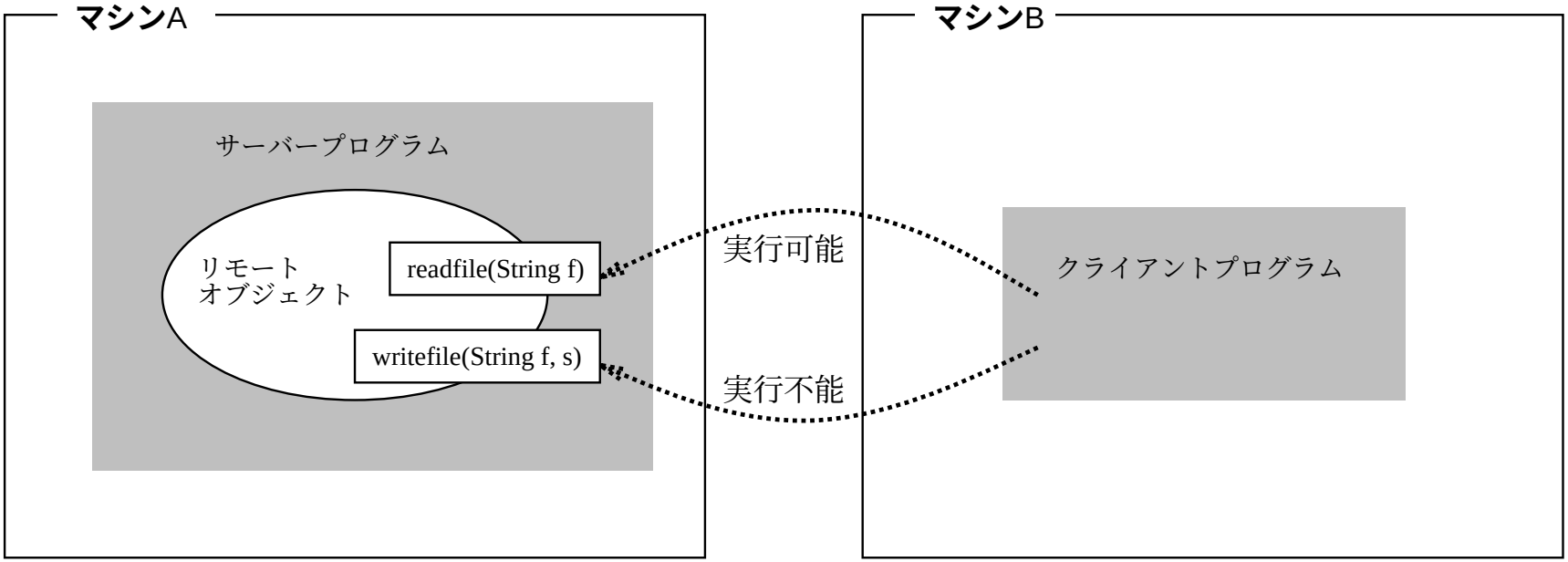
- ソフト部品の再利用促進にはその正確な仕様が必要.
- 従来の仕様は自然言語による説明程度.
- 形式的なものとしてはシグニチャや部品の内部状態に依存した条件の記述程度.
- 部品とその利用環境との関わりの重要性が増大: 同じ部品が異なる環境下で利用されることの増大, モバイルコードなど.

⇓

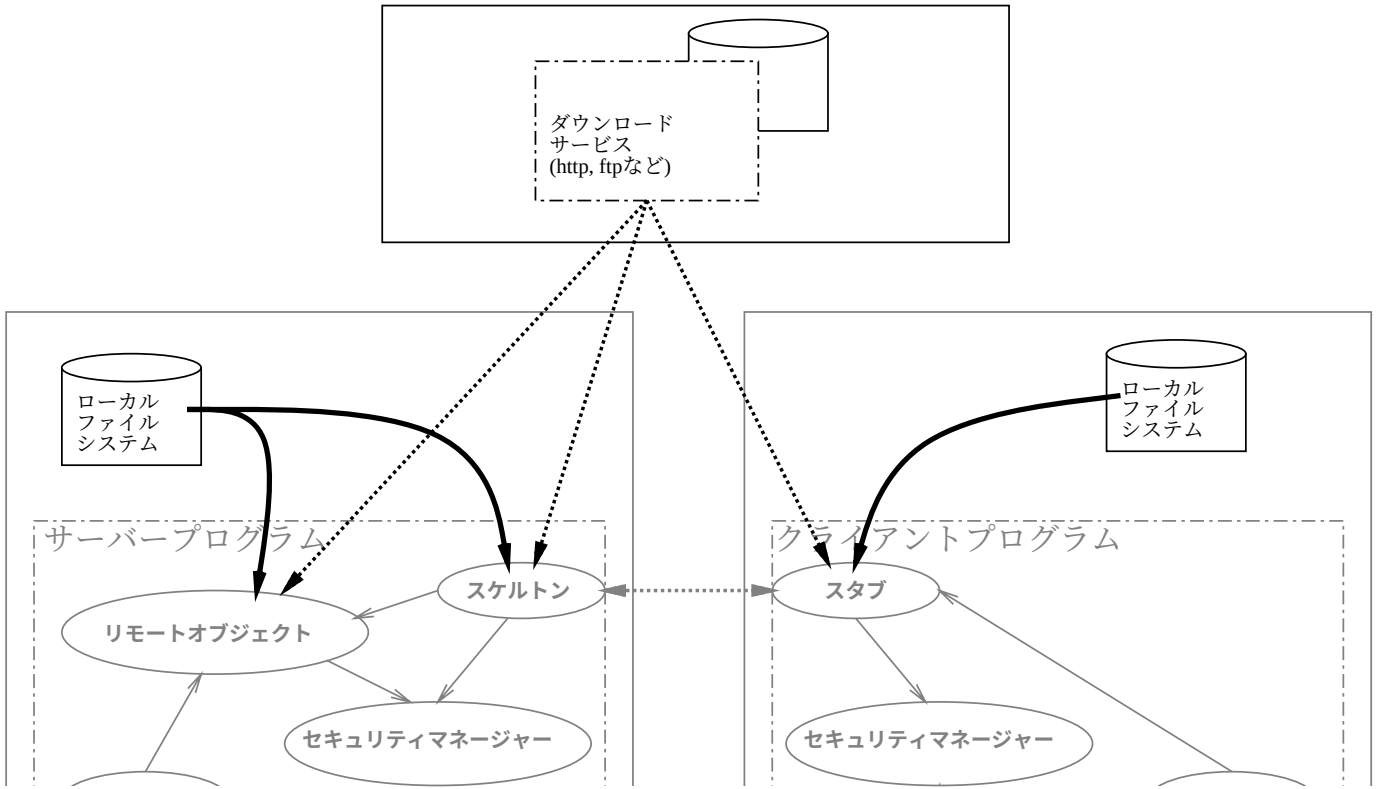
- 実行環境によって変化する部品の振る舞いを仕様化.
- 環境の記述方針に汎用性を持たせる

# 実行環境も含めた仕様化の例

サーバープログラム側のファイルを，リモート側から読めるが，書けないプログラムをセキュリティマネージャーを緩めることで実現する．



# 実行環境を多様化させる要因: 動的ロード機構




部品のコードが何処から VM にロードされたかで，その振る舞いが変わる，

振る舞いの制限はセキュリティマネージャー部品で定義される．

# 部品のソース

セキュリティマネージャー:

```
class MyManager extends RMISecurityManager{
  public synchronized void checkRead(String file){}
  public synchronized void checkWrite(String file){
    super.checkWrite(file);
  }
}
```

リモートオブジェクト:

```
public class MyRObject
 extends UnicastRemoteObject implements MyInterface{
  public MyRObject() throws RemoteException{ super(); }
  public String readfile(String s) throws RemoteException{
      return new String(SimpleIO.readfile(s));
  }
  public boolean writefile(String f, String s)
    throws RemoteException{
      return SimpleIO.writefile(f, s);
  }
}
```

# 部品の仕様 (1 / 2)

*MyManager*

$checkTable$ : 制限される動作 $\rightarrow \mathbb{B}$

*InitMyManager*

$MyManager'$

---

$checkTable'$ 読 $=$ true
$checkTable'$ 書 $=$ false
$\vdots$

# 部品の仕様 (2 / 2)

$$LoadPlace ::= Local \mid Network$$

**LoadMap**

$$impl, skel : LoadPlace$$

**pre.MyRObject.writefile**

$$MyRObject;\ MyManager;\ LoadMap$$

---

$$impl = skel = Local \lor$$
$$(impl = Network \lor skel = Network) \land chakTable \text{ 書}$$

**MyRObject.writefile**

$$f? : \text{ファイル名};\ s? : \text{ファイル内容}$$
$$\Delta MyRObject;\ \Xi MyManager;\ \Xi LoadMap$$

---

$$serverFS'\ f? = s?;\ \mathrm{dom}\ serverFS \cup \{f?\} = \mathrm{dom}\ serverFS'$$

# 適切な性質

ローカルファイルからスケルトンと実装をロードした場合は,

$$MyRObject.writefile$$
$$\mid impl = skel = Local \wedge serverFS\ f? \neq s?$$
$$\vdash serverFS \neq serverFS'$$

となり，ネットワークからどちらかをロードした場合は,

$$MyRObject.writefile$$
$$\mid skel = Network \wedge serverFS\ f? \neq s?$$
$$\vdash serverFS = serverFS'$$

となる.

## ここまでの記述の問題点

- 記述が煩雑である – writefile などの仕様.
- 記述方針に汎用性がない – メソッド毎にその性質を考慮して記述しなければならない.

# 記述の改善方針と解法

あらたなメソッドを記述する場合でも系統的に仕様を記述できるようにする

⇓

セキュリティ制限がどのような方針で行われているかを考慮しなおして，セキュリティマネジャーの仕様の記述を直す.

## セキュリティ制限の観点から資源を分類

RMI で操作する資源の型:
プログラム内の変数 or プログラム外のファイルシステム
or ネットワークポート

$$ResType$$
$$::= Internal \mid FileSystemR \mid FileSystemW \mid Port$$

どの型に属するかの値を個々の資源とともに保持することによって，セキュリティマネジャーの扱いを一般化.

# ResType に基づくセキュリティマネジャーの仕様

*AppletSecurityManager*

$$a : String \times ResType$$

$$\vdots$$

---

$$\forall x : String \bullet a = (x, Internal)$$

$$\forall x : String \bullet a \neq (x, FileSystemR)$$

$$\forall x : String \bullet a \neq (x, FileSystemW)$$

$$\vdots$$

このマネージェーでは x で総称される値が内部変数 (Internal) の場合のみ

# この仕様を利用した新たなメソッドの記述例

$$
\begin{array}{|l}
\hline
HelloImpl.sayhello \\
out! : String;\ rep! : Report \\
message : String \times ResType \\
SecurityManager \\
\hline
(out!, Strage) = message \\
SecurityManager[a/message] \Rightarrow \\
\quad rep! = OK \\
\neg SecurityManager[a/message] \Rightarrow \\
\quad rep! = Exception \\
\hline
\end{array}
$$

# Applet 下での推論例

$$HelloImpl.sayhello$$
$$[SecurityManager$$
$$/AppletSecurityManager]$$
$$\vdash$$
$$rep! = OK$$

# セキュリティマネジャーを変更した場合の推論例 (1 / 3)

ファイルシステムを読む許可を与えたセキュリティマネジャー

$FSReadSecurityManager$

$a : String \times ResType$

$\vdots$

---

$\forall x : String \bullet a = (x, Strage)$

$\forall x : String \bullet a = (x, FileSystemR)$

$\forall x : String \bullet a \neq (x, FileSystemW)$

$\vdots$

# セキュリティマネジャーを変更した場合の推論例 (2 / 3)

readfile メソッドの仕様

*HelloImp.readfile*

$f? : String;\ out! : String$

$serverFS : String \pfun Sting \times ResType$

$SecurityManager$

---

$\vdots$

$SecurityManager[a/serveFS\ f?] \Rightarrow$

$\quad (out!, FileSystemW) = serverFS\ f?$

$\vdots$

# セキュリティマネジャーを変更した場合の推論例 (3 / 3)

FSReadSecurityManager では，readfile は実行可能であるが，

$$\begin{array}{l} HelloImp.readfile \\ \quad [SecurityManager \\ \quad /FSReadSecurityManager] \end{array} \vdash rep! = OK$$

Applet では実行可能でない.

$$HelloImp.readfile\ [SecurityManager/AppletSecurityManager] \vdash rep! = Exception$$

# まとめ

- 部品の形式仕様をその実行環境を考慮して与えた.
- その記述方法にも，ある程度の汎用性を与えた.