モバイルコード技術とセキュリティポリシィ (2)

2005年12月21日

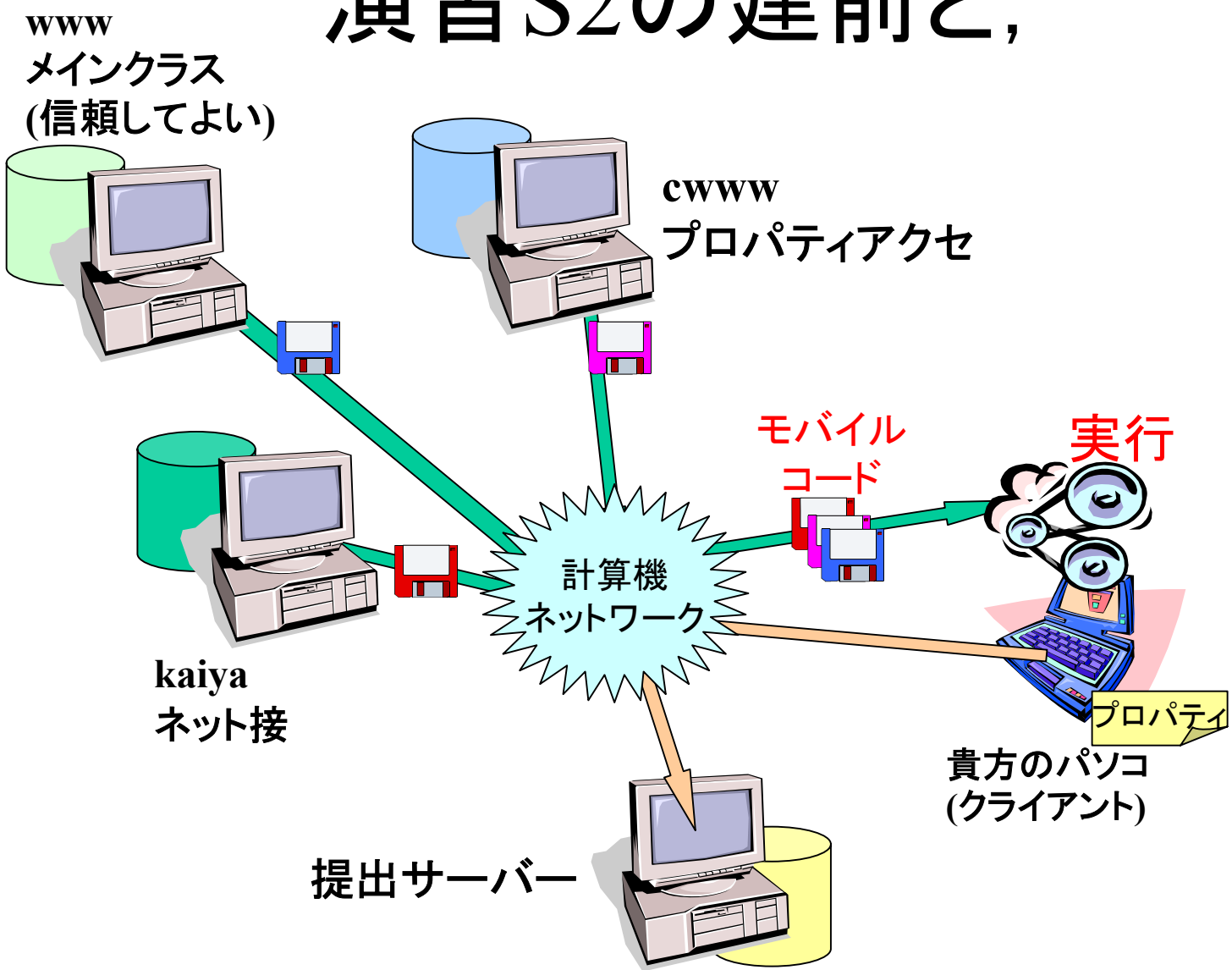
海谷 治彦

複数モバイルコードの利点・欠点

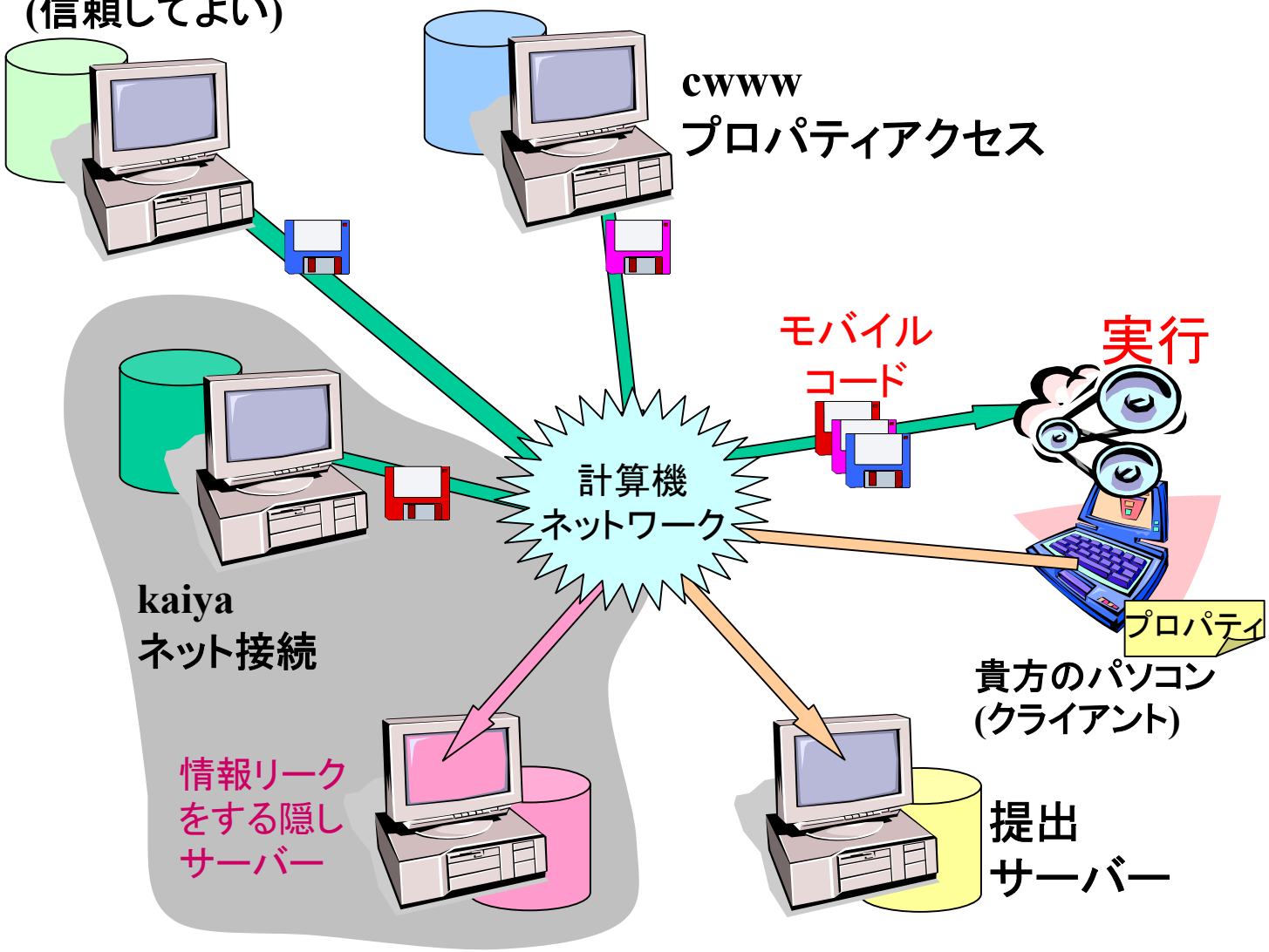
- アプリより細かい粒度(コンポーネント)でのソフトウェアサービスを「比較可能な商品」として流通させられる.
  - 同じインタフェースを持ち，性能や効果，値段が違うコンポーネントを取捨選択して利用可能.

- セキュリティポリシィの決定がより複雑になる.

演習S2の建前と,


www
メインクラス
(信頼してよい)
cwww
プロパティアクセ
モバイル
コード
実行
計算機
ネットワーク
kaiya
ネット接
プロパティ
貴方のパソコ
(クライアント)
提出サーバー

演習S2の現実 (漏洩あり)



演習S2で要求されていた事項

- user.name, user.dir プロパティを読み出せる.
- 実行時に指定されるサーバーに読み出したデータを送信する.

演習S2のポリシーファイル

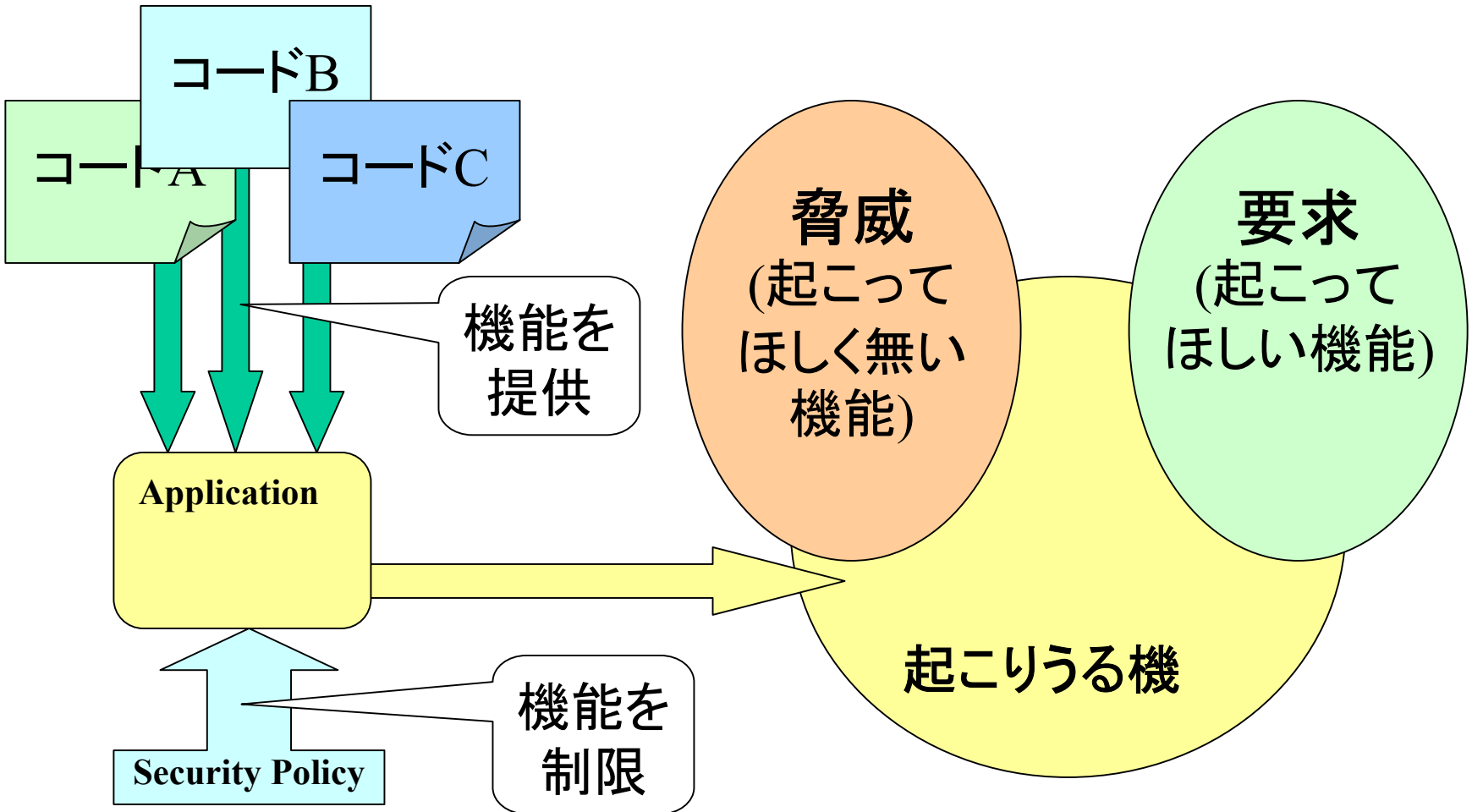
```
// 前略

// for main method invocation
grant codeBase "http://www.cs.shinshu-u.ac.jp/~kaiya/java/tmp/" {
    permission java.security.AllPermission "", "";
};

// server connection
grant codeBase "http://kaiya.cs.shinshu-u.ac.jp/tmp/" {
    permission java.net.SocketPermission "*.cs.shinshu-u.ac.jp", "connect";
};
// property access
grant codeBase "http://cwww.cs.shinshu-u.ac.jp/~kaiya/" {
    permission java.util.PropertyPermission "user.name", "read";
    permission java.util.PropertyPermission "user.dir", "read";
};
```

演習S2のポリシーで可能なこと

- user.name, user.dir プロパティを読み出せる.
- 実行時に指定されるサーバーに読み出したデータを送信する.
- プログラムが読み込めるデータを任意のサーバーに送信できる.

  ・・・

起こりうる機能，脅威，要求

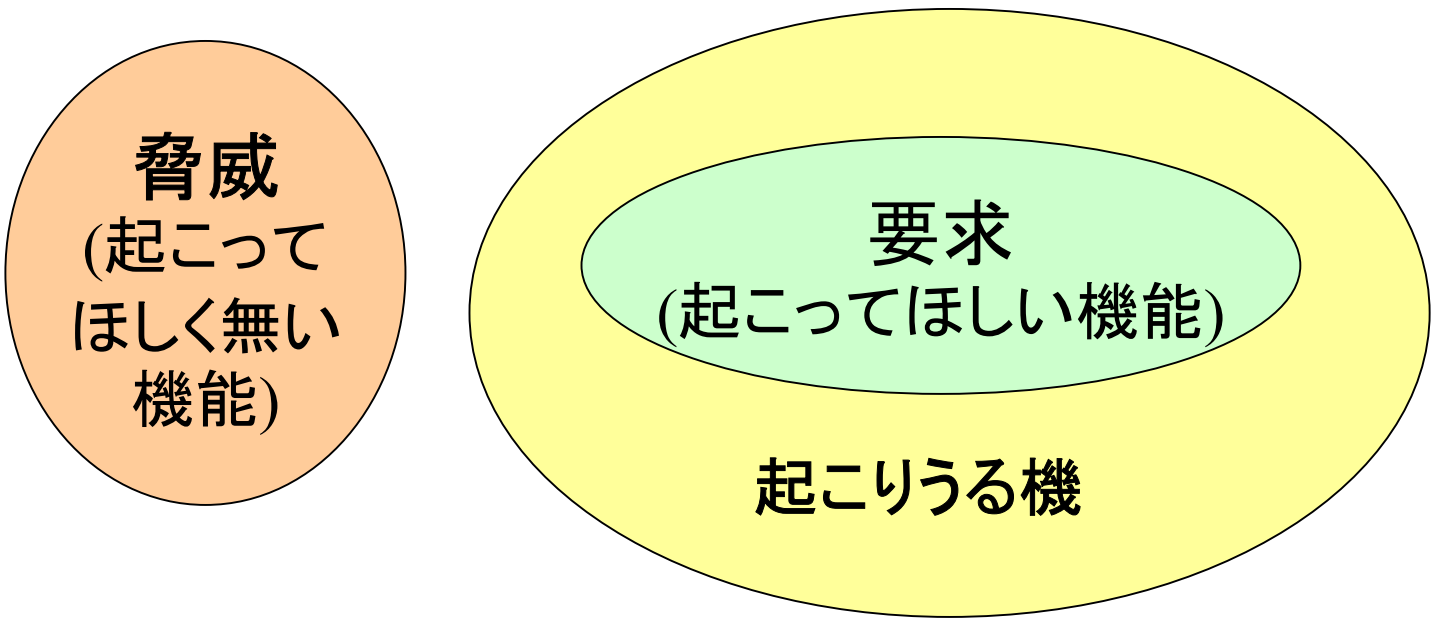


ポリシーによるコードの再利用

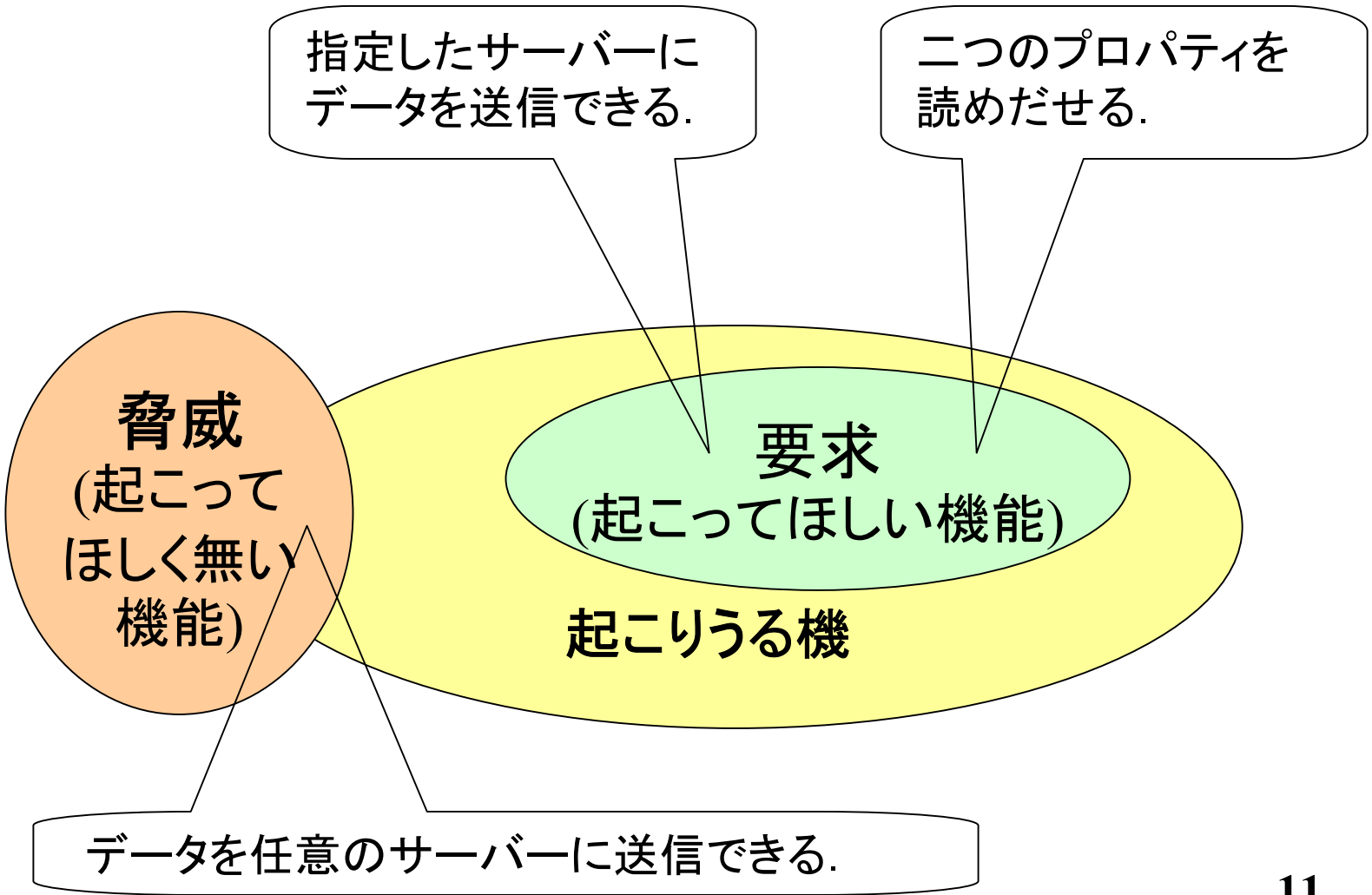
- モバイルコードに限らず，コード(クラスや関数)をブラックボックス的に利用すること.
- 前述のスライドのように，一般に要求と起こりうる機能は一致しない.
- さらに悪いことに，脅威(起こってほしく無いこと)を許すことになるかもしれない.
- 要求 ∩ 脅威 = Φ は常に正しいだろう.
  - じゃないと矛盾してる.

理想的な設定

- 下記になるのが理想.
  起こりうる機能 ⊇ 要求
  起こりうる機能 ∩ 起こりうる機能 = Φ

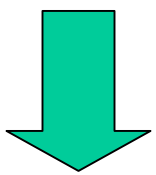


演習S2の場合の実際


指定したサーバーに
データを送信できる.
二つのプロパティを
読めだせる.
脅威
(起こって
ほしく無い
機能)
要求
(起こってほしい機能)
起こりうる機
データを任意のサーバーに送信できる.

脅威を完全に抑制できない原因

- 実行前の情報の不確定性
  - 演習S2はこのケースか.
    (サーバー名は実行後に決まる)
- Javaのアクセスコントロールの特性から
  - 詳細は次回に.
- 起こりうる要求の見落とし
  - ポシリーを緩めにしてしまったとか・・・



- 完全に脅威を抑制するのは難しい！

「妥協」の必要性

- 妥協
  1. 要求の達成を一部「断念」する.
  2. 脅威の可能性を一部「黙認」する.
- どちらにしろ，どんな妥協を行っているかを認識しないでソフトウェアを実行するのが一番アブない.
- 演習S2の解答例では，
  - 「任意のサーバーへの接続」を黙認
  - 「正規のサーバー指定を柔軟に行う」要求を充足.

  させたポリシー設定である．逆に，
  - 「特定サーバーの事前指定」，すなわち柔軟設定を断念.
  - 「任意のサーバーへの情報漏洩」の脅威を回避.

  という選択もアリ.
- ・・・ってことで，来週以降は「妥協」の明確化について演習を進めます.