アスペクト指向ソフトウェアアーキテクチャスタイルの構築

The Construction of an Aspect-Oriented Software Architecture Style

熊崎 敦司*　野呂 昌満†

Summary.
We propose software architecture style to unify aspects and collaboration fields. The proposed style gives software engineering guidelines for appropriate usage of the message hook.

1　はじめに

横断的コンサーン分離問題（the problem of separation of crosscutting concerns）[2] の解記述およびロールモデルで解を記述することが POP 技術の目指すものと考える．これらの解記述のガイドラインを与えることで，一定レベル以上の能力を持つ技術者ができるだけ一意の解を記述できることを目標とする．本研究では，メッセージフック（合流点にメッセージ通信コードを付加するというアスペクト指向プログラミング言語 [3] のアドバイス機構の基本的枠組み）で，アスペクト [2] と協調場（collaboration field） [6] を統一的に取り扱う．そのさいの関係に制約を定義し，アスペクト指向ソフトウェアーキテクチャスタイルとして提案する．この制約を守ることが解記述のガイドラインとなる．

2　アスペクト指向ソフトウェアアーキテクチャスタイル

コンサーンや協調場の集合としてソフトウェアを分割するさい，これらの階層化が必要だと考える．階層の末端となるコンサーンや協調場の実現として導出した構成要素はオブジェクトであり，システムはオブジェクトの集合として構成される．3 節の事例において，この考えが妥当なものであることを示す．

2.1　構成要素と関係の定義

提案するアスペクト指向ソフトウェアアーキテクチャスタイル（AOSAS）の構成要素はアスペクト，ロール，オブジェクトである．各構成要素を VDM-SL の型定義を用いて記述すると図 1 のようになる．各構成要素は名前とインタフェースを保持する（図 1 中 2〜6，12，17 行目：以下括弧内の行数は図 1 中の行数を示す）．

提案する AOSAS において，コンサーンや協調場を統一的に取り扱うために，これらが規定する構成要素をアスペクトとする．コンサーンや協調場の階層を自然に表現するために，アスペクトを階層的に記述する．ソフトウェアアーキテクチャは階層的に記述されたアスペクトの集合となる．アスペクトは内部にロールの集合を保持する（7，15 行目）．ロールはアスペクトまたはオブジェクトの仕様であり，アスペクトまたはオブジェクトと結合する[1]（10，13，20 行目）．これにより，アスペ

*Atsushi Kumazaki, 南山大学数理情報学部情報通信学科

†Masami Noro, 南山大学数理情報学部情報通信学科

[1]Java の intarface と実現の関係に相当する．二つのものが合わさり，一つのものとして動作する．ロールを通して，アスペクトとオブジェクトを統一的に取り扱う．

```
 1: types
 2: Component :: name : CName
 3:              interfaces : set of Interface;
 4: CName = token;
 5: Interface = token;
 6: Aspect    :: com : Component
 7:              roles : set of Role
 8:              usesRelation : map CName to Aspect
 9:              is-aRelation : map CName to Aspect
10:              linked : Role
11: inv a == a.linked in {rng(a2.linked.roleRelation)|a2 in rng(a.useRelation)} &...
12: Role      :: com : Component
13:              link: Object | Aspect
14:              roleRelation : map CName to Role
15:              partOf : Aspect
16: inv r     == r.partOf = r2.partOf & r2 in rng(r.roleRelation)
17: Object    :: com : Component
18:              messageRelation : map CName to Object
19:              messageHookRelation : map CName to Object;
20:              linked : Role;
21: inv o == o.linked in {rng(o2.linked.roleRelation)|o2 in rng(o.messageRelation)} &...
22:          o.linked.partOf = o2.linked.partOf & o2 in rng(o.messageHookRelation) &...
```

図 1　VDM-SL による構成要素の定義

クトの階層を記述できる．これらのロール（同一アスペクト内のロール）は互いに関連することができる（14，16 行目）．ロール間の関連は，それらを規定したコンサーンが横断的に関連すること，協調場毎に役割を変えるオブジェクトが存在することを表す．それぞれに結合するアスペクトまたはオブジェクト間に関係を定義できることも意味する．

アスペクトは，コンサーン，または協調場を基に分割されたソフトウェアの一部を表すので，これらを合成してシステムを構築する．一般にコンサーンの分離による分割には，AspectJ [3] 分割[2]と Hyper/J [5] 分割[3]がある．AspectJ 分割によって分割されたアスペクト間には同一のロールが存在しないが，Hyper/J 分割によって分割されたアスペクト間には同一のロールが存在する．以下の 2 つの関係によりこれらの関係を自然に記述する．協調場 [6] 分割は一つのオブジェクトが持つ複数の役割を分割するので，Hyper/J 分割と同様に扱う．

- use 関係（8 行目），一方のアスペクトが保持するロールが，他方のアスペクトが保持するロールを知っているという関係である．抽象クラスが属性として他の抽象クラス型の変数を保持するのに相当する．ただし，結合度は低い．
- is-a 関係（9 行目），それぞれのアスペクトが保持するロール間に is-a 関係（ロールはインタフェースを持っているので，Java の intaface 同士の extends 関係に相当する）が存在する関係である．子アスペクトは親アスペクトの関係を引き継ぐ．

これらの関係が定義できるのは，それぞれの結合先のロール間に関連がある場合に限られる（11〜12 行目）．

提案する AOSAS では，これらの分割されたアスペクトの合成を統一的に行う．構

[2]分離したコンサーンが規定する構造が，コアコンサーンが規定する構造とは独立となる分割．

[3]分離したコンサーンが規定する構造が，コアコンサーンが規定した構造に従う分割．

成要素の分割にともない，前述の 2 つの関係はともに後で説明するメッセージフックの関係に洗練されるので，両者の分割を区別なく扱うことができる．

システムは終端のアスペクト内のロールと結合したオブジェクトの集まりである．オブジェクト間には以下の関係を定義することができる．これらの関係はロール間の関係，またはアスペクト間の関係から導出される（21，22 行目）．

– （一般のオブジェクト指向の）メッセージ通信関係（18，21 行目）
– メッセージフック関係（19，22 行目）

異なるアスペクトに属する（結合先のロールが異なるアスペクトに保持される）オブジェクト間の結合度を弱めるためにメッセージフック関係を用いる．

2.2 関係記述における制約

提案した AOSAS は関係の記述に制約を与えている．この制約に従えば，適切にメッセージフックを用いることができるので，この制約を守ることがガイドラインとなる．提案した AOSAS が示す制約を以下に説明する．

1. アスペクトの分割にともない，ロール間の関係[4]をアスペクト間の関係（use 関係，または is-a 関係）に洗練する．AspectJ 分割の場合は use 関係に，Hyper/J 分割，協調場分割の場合は is-a 関係に洗練する．結合先のロール間に関連がない場合はそれぞれのアスペクト間に関係を記述することはできない．
2. アスペクト間に関係が存在する場合，その関係は，それぞれのアスペクトに属するオブジェクト間のメッセージフック関係に洗練する．それぞれのオブジェクトが属するアスペクト間に関係がない場合，また，それぞれのオブジェクトが同一のアスペクトに属する場合に，オブジェクト間の関係がメッセージフックとなることはない．
3. ロール間の関係をオブジェクト間のメッセージ通信関係に洗練する．同一のアスペクトに属するオブジェクト間の関係はメッセージ通信関係となる．メッセージ通信関係が記述できるのはこの場合に限られる．

3 アプリケーションプロトコルソフトウェアへの適用と考察

提案した AOSAS をアプリケーションプロトコルソフトウェアに適用し，ソフトウェアアーキテクチャを構築する．本稿では，提案した AOSAS が適用可能なことを示すことを第一義として，アプリケーションプロトコルソフトウェアにおけるコンサーンや協調場の一般性および妥当性に関する議論は省略する．

階層モデルを用いてアプリケーションプロトコルソフトウェアの構造を整理してきた経験 [4] から，並行処理，実時間処理，例外処理，セキュリティのコンサーンがコアコンサーンと横断的に関連していると認識した．さらに，コアコンサーンをプロトコル処理と考え，使用者入力処理，状態遷移処理，サーバ結果処理，トランスポートプロトコル処理，ネットワークプロトコル処理の協調場が抽出できた．この場合，役割が変わるオブジェクトはパケットである．これらのコンサーン，協調場の階層を基に，提案した AOSAS を適用すると，図 2 に示すソフトウェアアーキテクチャが得られる．

これまでにソフトウェアアーキテクチャの記述支援としてアーキテクチャ記述言語 [1] が提案されているが，関係の記述に制約を与える試みはされていない．本研究

[4] 横断的に関連するコンサーンに対応するロール同士は関連する．

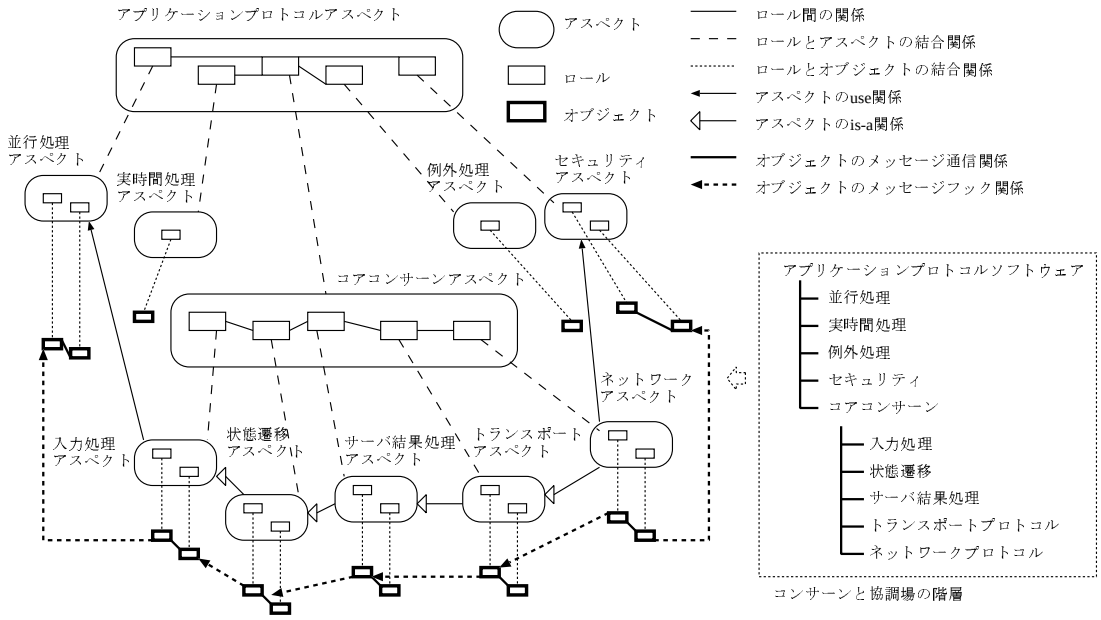

図 2　アプリケーションプロトコルソフトウェアのソフトウェアアーキテクチャ（詳細は省略）

では，ソフトウェアアーキテクチャ記述時の関係の記述に制約を与えることで，記述者の意思決定を支援している．その結果，一定レベル以上の能力をもつ技術者は適切にメッセージフックを使用することが可能となったと考えられる．

4　おわりに

提案した AOSAS において構成要素間の関係記述に制約を課し，ソフトウェアアーキテクチャ記述のガイドラインとした．提案した AOSAS に従えば，AspectJ 分割，Hyper/J 分割，および協調場分割した構成要素を合成して，システムを構成することになる．提案した AOSAS では，これらの合成を統一的に取り扱っている．今後は，簡素で自然な記述支援のために AspectJ 分割と Hyper/J 分割の統一的な取り扱いを目指す．

謝辞　この研究は，平成 16 年度パッヘ研究奨励金 I-A-I,II の助成を受けて行った．

参考文献

[1] R. Allen and D. Garlan, “Formalizing architectural connection,” *Proceedings of ICSE*, 1994
[2] T. Elrad, M. Aksits, G. Kiczales, K. Lieberherr and H. Ossher, “Discussing Aspects of AOP,” *Communications of the ACM, 44, 10* , 2001.
[3] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm and W.G. Griswold,“An Overview of AspectJ,” *Proceedings of ECOOP*, Springer-Verlag. Hungary, 2001.
[4] A. Kumazaki, M. Noro, H. Chang and Y. Hachisu, “An application Framework for TCP/IP Applications,” *Proceedings of COMPSAC*, pp.627-634, 2002.
[5] H. Ossher and P. Tarr, “Multi-Dimensional Separation of Concerns and the Hyperspace approach,” *Proceedings of SSACT*, 2001.
[6] T. Tamai, “Evolvable Programming based on Collaboration-Field and Role Model,” *Proceedings of IWPSE,* 2002.