[評] 著者らのハイパーメディアでのフレームワークを拡張して，文書の執筆の共同作業の支援システムを構築しているが，この二つは非常に相性がよいと思われる．これからは，この分野の研究がおおいになされるであろう．特に，warm linking の技術は何か他にも応用分野があるのではないか，と思われる．

（三菱電機(株)情報電子研究所 阿倍博信）

90-42 形式的手法にまつわる7つの神話

A. Hall : Seven Myths of Formal Methods

[*IEEE Software*, Vol. 7, No. 5, pp. 11-19 (1990)]

Key : Formal methods, formal specification, Z.

IEEE では，*IEEE Computer*, *IEEE Software*, *IEEE Trans. on Soft. Eng.* の三誌の 1990 年 9 月号を使って，形式的手法 (formal methods) に関する三部作を組んでいる．なかでも，後者二誌は，このテーマに絞った特集号という力のいれようである．この三部作の構成については，*IEEE Software* 中の Guest Editor's Introduction[1] に詳しく述べられている．

形式的手法というと，仕様記述言語Z，ソフトウェア仕様記述法 VDM 等のようにヨーロッパでの活動が盛んである．特にイギリスでは，Z の Spivey, CSP の Hoare を擁するオックスフォードを中心として，研究活動にとどまらず，実際の応用面でも多くの事例が報告されている．今回の三部作でも，普段よりもイギリスからの論文が目立っている．

本論文は上記 1) のすぐ後に登場する論文である．著者はイギリスのソフトウェア会社に勤めるコンサルタントで，実際のプロジェクトで形式的手法を使用している．まず最初に，自らの経験に基づき，形式的手法に関して一般に広まっている神話を次にあげる7つに分類している．

1. 形式的手法は，ソフトウェアが完璧であることを保証してくれる．
2. そもそも形式的手法とは，プログラムの証明に関することである．
3. 形式的手法とは，安全性が重視されるシステムにおいてのみ有用である．
4. 形式的手法を使うには，高度な訓練を受けた数学者が必要となる．
5. 形式的手法は開発経費を増大させる．
6. 形式的手法は顧客にとって受け入れ難いものである．
7. 形式的手法は，現実的な大規模なソフトウェアには適用されていない．

この中には，形式的手法に対して批判的なものもあれば，1のように過大評価ととれるものも含まれている．

次に，著者が実際に携わった CASE 関係のプロジェクトを例にとり，7つの神話を一つずつ吟味していく．CASE プロジェクトの他にも，IBM, テクトロニクス，ロールス・ロイス社等の例を適宜取り入れている．特にロールス・ロイス社の例では，上記の 5. に関連して，仕様作成に要した時間のうちの7パーセント分のおかげで，プロジェクト後期で大きな経費節減をはかることができたという事例をあげている．

本論文は，形式的手法は決して万能ではないが大変に強力なツールであり，一般の開発担当者はもっと良くそれを理解するべきであると結論付けている．また，上記の7つの神話のそれぞれに代わるものとして，次の7つの事実をあげている．

1. 形式的手法は，開発初期において誤りを発見するのに大いに役立つ．また，ある種の誤りに関してはそれをほとんど除去することが可能である．
2. 形式的手法とは，開発者自身が，自分の開発しようとしているシステムについて詳しく検討することによって機能するものである．
3. 形式的手法は，ほとんどどの応用分野でも有効である．
4. 形式的手法は数学的仕様に基づいているが，これはプログラムよりずっと分かりやすいものである．
5. 形式的手法は開発経費を低減させる．
6. 形式的手法は，顧客が何を購入しようとしているのかを理解する手助けとなる．
7. 形式的手法は，現場での現実のプロジェクトに適用され成功をおさめている．

[評] 解説も平易で啓蒙的な論文である．Zによる仕様の簡単な例もあり，詳しい知識がなくとも十分に理解できる．7つの神話に関する議論の内容は主に定性的なものであるが，十分に説得力があり，ロールス・ロイス社の例などは，さすがに本場のイギリスといった印象を受ける．この種の論文は一方的な賛美に終始しがちだが，単に形式的手法の利点を述べるだけでなく，その限界も明確に示している点は好感がもてる．

参 考 文 献

1) Gerhart, S. L.: Applications of Formal Methods: Developing Virtuoso Software. *IEEE Software*. Vol. 7, No. 5, pp. 7-10 (1990).

(静岡大学教育学部 白井靖人)

90-43 拡張可能共有メモリ・マルチプロセッサ: DASH

Daniel Lenoski, Kourosh Gharachorloo, James Laudon, Anoop Gupta, John Hennessy, Mark Horowitz, and Monica Lam.: Design of Scalable Shared-Memory Multiprocessors: The DASH Approach

[*Proc. of Compcon* '90 (Feb. 26-Mar. 1 1990)]

Key: Scalable shared-memory multiprocessors, hardware-supported coherent caches, distributed directory-based protocol.

DASH は，共有メモリ・マシンのプログラミングのしやすさと，メッセージパッシング・マシンの拡張性を兼ね備えることを目的として，スタンフォード大学で現在研究されている拡張可能共有メモリ・マルチプロセッサ・システムである．DASH はネットワークで接続された共有メモリ・マシン間のキャッシュの一致性をハードウェアで維持することで，共有データのアクセス遅延をおさえ，プログラミングを容易にする．本論文では DASH のキャッシュの一致性の維持機構，及びメモリ遅延を減少し，効率のよい同期を提供する機構について述べている．また，試作機の構成の概要についても述べている．

現在の共有メモリ・マシンは，ネットワークによる拡張性を持たないか，拡張性を持つがキャッシュの一致性をハードウェアで維持しないという弱点を持つ．前者の例としては Encore Multimax, Sequent Symmetry などのバス結合マシンが，後者の例としては BBN Butterfly, IBM RP3 などのネットワーク結合マシンがあげられる．

DASH は，ネットワーク結合マシンの各プロセッシング・ノードにディレクトリを導入して，キャッシュの一致性をハードウェアで維持しようとするものである．さらにネットワークを介したアクセスの遅延を減少するために，新しいキャッシュの一致性と，特別な命令を導入してパフォーマンスを高く維持している．

DASH のアーキテクチャはプロセッシング・ノード（クラスタ）を複数，ネットワークを通じて接続したものである（図）．各クラスタは各々キャッシュを持つ高性能プロセッサと共有メモリをスヌープ・バスで接続したものである．クラスタはディレクトリ・コントローラを介してネットワークに接続される．

DASH ではディレクトリ・コントローラによってクラスタ間のキャッシュの一致性をハードウェアで維持する．各クラスタの共有メモリは各々固有のアドレスに置かれている．それらはディレクトリ・コントローラによってキャッシュ・ライン長ごとに各クラスタにおけるキャッシング状態が管理されている．プロセッサからのリード／ライト・リクエストはクラスタ内の

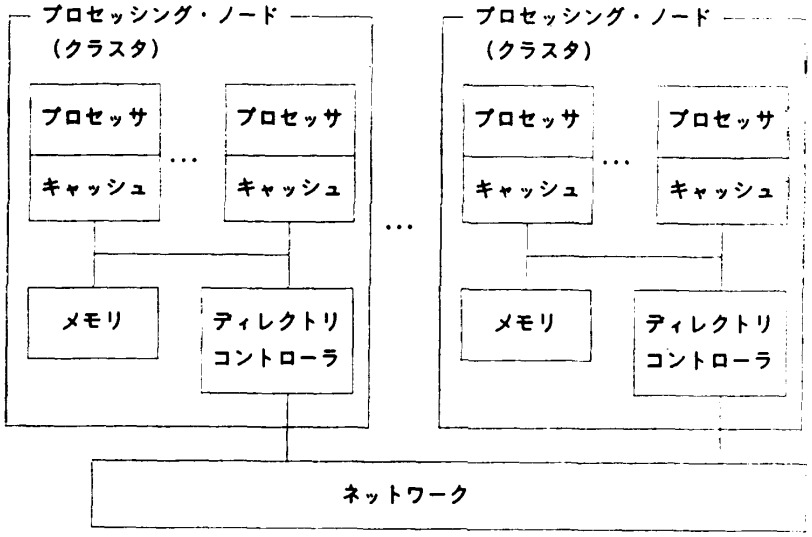

図 DASH アーキテクチャ