30. ソフトウェアシステムの安全性とドメイン制御機構

東京芝浦電気株式会社　星野康夫・江川明人
増田龍彦

0. はじめに

表題のソフトウェアシステムの安全性ということばはかなり広範囲を対象としてしまうが，ここではドメイン制御という方法に重点を置いて，それをシステムアーキテクチャ上にどのように実現できるかを述べる。

ドメインとは，アクセス可能性の範囲，即ち，メモリやファイルなどの利用できる対象領域群の場所と大きさ，それらの論理的な単位に対する読み・書き，実行・呼出し権限などのアクセス権のワンセットのことである。一つのプロセスは，実行中のどの時点に於ても必らずいずれかのドメインの中で動作しており，そのドメインに許された範囲内でのアクセス権の行使のみ許される。従って一つのドメインから他のドメインへ制御が移る際には，そのプロセスのアクセス権は動的な意味で変化する。

このように，ドメインの概念は，ソフトウェアあるいはハードウェアという区別とは独立の概念であるが，本稿では，ソフトウェアシステムの実行の際に，プロセスの誤動作によるプログラム破壊，あるいは制御逸脱などを，論理的単位でできるだけ防止するためのドメインの構成方法と，それにマッチした機構のいくつかを紹介することにしたい。

1. 動的保護

1.1 従来の動的保護

従来，ソフトウェアが，実行中に犯す誤りや破壊から免がれ，あるいはそれらの誤動作を効果的に検出して，ソフトウェアシステムの実行時の安全性を確保するために次のような方策がとられてきた。

レベル1. オペレーティングシステムの保護
管理プログラムが動作・アクセスできる空間と，被管理プログラムのそれとを，プロセッサ・モードに特権モードと非特権モードを設けることにより区別する。

レベル2. プロセスの保護
被管理プログラム間での動作・アクセスできる空間の分離のために，バウンダリ・レジスタなるものが設けられ，一つのプロセスによる動作・アクセスが，間違って他のプロセスに及ぶことを防止する。

レベル3. 手続きの保護
一つのプロセス空間内で，さらに各手続きの空間を分離して，一つの手続きが誤って他の手続き内を参照あるいは破壊したりしないようにする。これは，いわゆるセグメンテーションによりソフトウェアやハードウェアにより考慮され実現されている。

レベル4. セグメントの保護
一つの手続き内の命令部分とデータ部分を区別して動作空間とアクセス空間を明確に分離する。これはセグメンテーションによる一般的な形態

であり，再入可能プログラムの作成の場合にもとられる形式である。

以上第4のレベルまでは空間の分離という意味で一般的であり，一部はハードウェアにより，他のほとんどはソフトウェア上の約束により実現されている。しかしながら，システムアーキテクチャそのものの中に，制御の逸脱，アクセスの暴挙をきめ細かく禁止できるような機構を持っていない限り，プログラミング上の約束だけでは各プログラムが自分自身の命令コードやデータを確実に守ることはできないし，自身の間違いを確実に防ぐこともできない。

1.2 より繊細な動的保護

ソフトウェアシステム上で発生する，アクセスや制御の暴走は何に起因するのかと言うと，もちろんソフトウェアの論理の誤りによるものが大多数であるが，その結果目的領域の境界を超えてアクセスしたり，目的地点とは違った場所へ制御を渡したりしてしまう動作によるものである。これらを観察するに，まず，命令コードとデータが隣接ないしは明確に区別されず混在していること，またアドレスデータが通常のパターンデータと同等に取り扱われ，それ自体でも領域を示すアドレス，制御を移す点を示すアドレスなどの明確な区別がないまま混在して用いられることなどに大きな原因があると思われる。そこで 1.1 節で述べた空間の分離の考えに，さらに次のようなレベルでの保護機構をシステム的に実現することが，ソフトウェアシステムの実行時の安全性を増すために役立つと考えられる。

レベル5. アドレス・データ空間の保護
データ空間をまずアドレス・データ空間とパターン・データ空間に分離すること。そしてそれぞれの空間をアクセスする命令を別々に設けることにより，それらの不用意な混同を自動的に検出できるようにすること。

さらに，これより下位のレベルでは空間分離ではなく，レベル5で述べたアドレス・データ空間に入るべき要素，すなわちアドレス・データの分類である。

レベル6. 領域アドレスと制御点アドレス
アドレス・データは空間の任意の部分をわく取りするようなものと，プログラムの制御の流れを移す地点を示すものとが区別されること。

レベル7-1. 入口点アドレスと戻り点アドレス
制御点アドレスをさらに，副プログラム呼び出しの入口点を示す入口点アドレスと，呼び出し点の直後に戻らずに好みの地点へ制御を戻すための戻り点アドレス・データに区別する。

レベル7-2. アドレス領域アドレスとデータ領域アドレス
領域を表示するアドレス・データを，その領域に格納できるデータがアドレスかパターンかで区別して，レベル5でのアドレス・データ空間とパターンデータ空間の分離を完成させる。

以上では「空間」にポイントを置いていくつかのレベルでの種々の区別を考察してきたが，これらの空間のすべてが静的なものに限られるのではなく，プログラムの操作により動的に変化し得るものである。従ってレベル7-1の戻り点アドレスのようなものでは，再帰的プログラム手法などとの関連から「時間」的要素も含まれてくる。

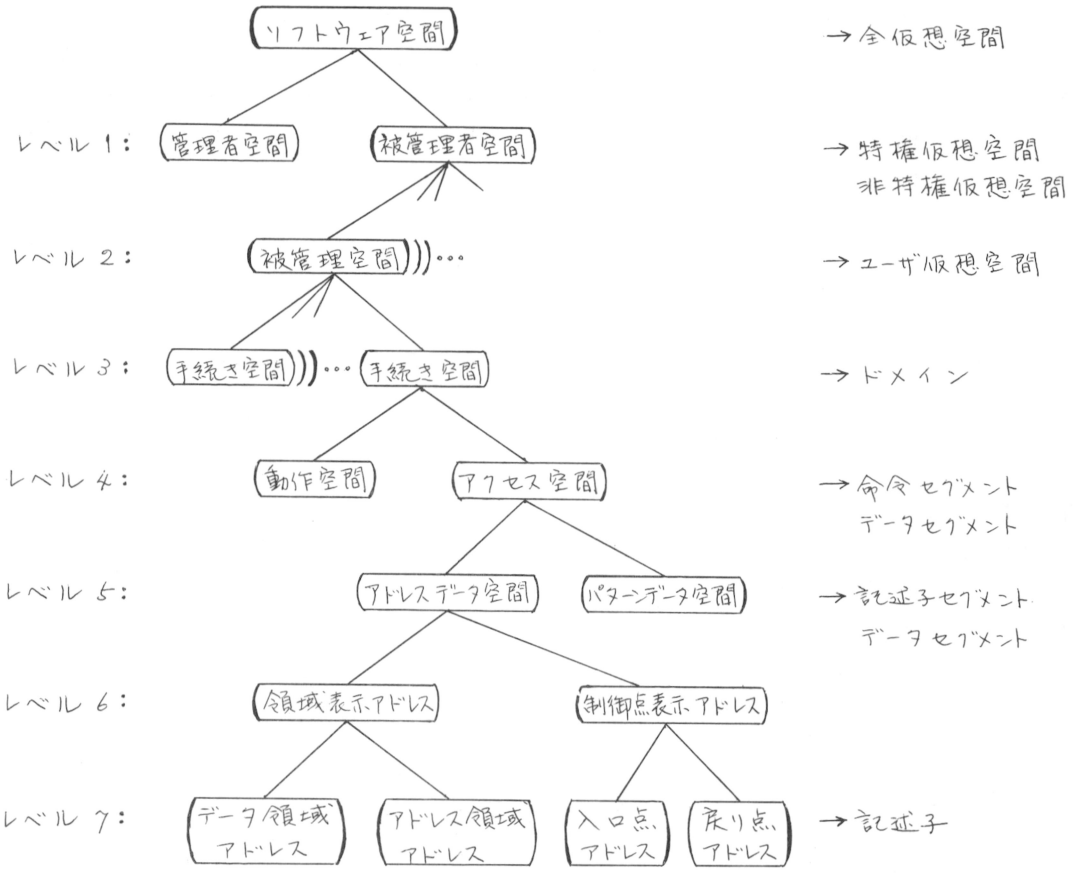


図-1　ソフトウェア空間のレベル分け

2. アドレス・データの表現

前節に述べたところから，アドレスデータの表現には単なる一次元的情報では不十分なことがわかる。そこで，アドレス情報を完全に記述し，アドレス・データ・空間にのみ格納できる，以下のようなアドレス記述子を設ける。

A. セグメント記述子（SD）

パターン型データ領域のベース・サイズ，読み出し・書き出し・実行などの許可，そして特権性およびこの記述子の複写許可などの情報を持つ。

B. 記述子セグメント記述子（DD）

アドレス型データ領域を表現するもので，SDと同様の情報を含む。

C. 入口記述子（ED）

保護・制御範囲（ドメイン）の異なる手続きへの入口点アドレスを表現する。目的ドメイン，入口点を含むコードセグメントとその中のオフセットを示す情報，そして自身の複写許可フラグを持つ。

D. ラベル記述子（LD）

いわゆる非正規復帰を実行するために，動的に作成されるべき記述子で，

戻りドメインを示すためにスタックに関連する時間的情報を持つ以外は入口記述子と同じ情報を含む。

これらの記述子に対して，その示す許容範囲を縮小するための命令を用意し，許可フラグのOFFや，サブセグメントの記述子を作り出すことができるようにする。この縮小機能は自ドメイン内のセグメントの小部分をパラメタとして他ドメインの手続きに渡すときに自ドメインの防御に必要である。

3. ドメイン

アドレス・データを記述子という形で識別できるようにしたことにより，各手続きをそれぞれ別々のドメインに構成して，1章で述べたレベル3の空間保護を確立することが可能になる。即ち各記述子が，それぞれどのような形態のアクセス権をどの領域あるいは対象について持っているかを示しているので，各ドメインはそれぞれの含む記述子の集合によって表現することができる。

3.1 静的ドメイン

FORTRAN，COBOLなどの高級言語の1翻訳単位の含む要素を，1章レベル4,5の保護対象に分類してみると

A. 命令コード・セグメント
B. データ・セグメント
C. 他手続きへの入口点参照
D. アドレス・データ・セグメント

Dは他手続きから受け取った記述子や，複写した記述子などの作業領域。

以上の4種類が，標準的な手続きの含む静的に決定されたアクセス権の及ぶ範囲となる。これらを表現する記述子のすべてをまとめて，連結セグメントと呼ばれる特別の記述子セグメントを設ける。アドレス・データの取り出しを，このセグメント内のオフセット指定でのみ行えるようにシステムを構成すれば，手続き自身が間違っても連結セグメントの外をアクセスすることができないので他ドメインを守ることになり，結局他ドメインからの自ドメインの保護となる。

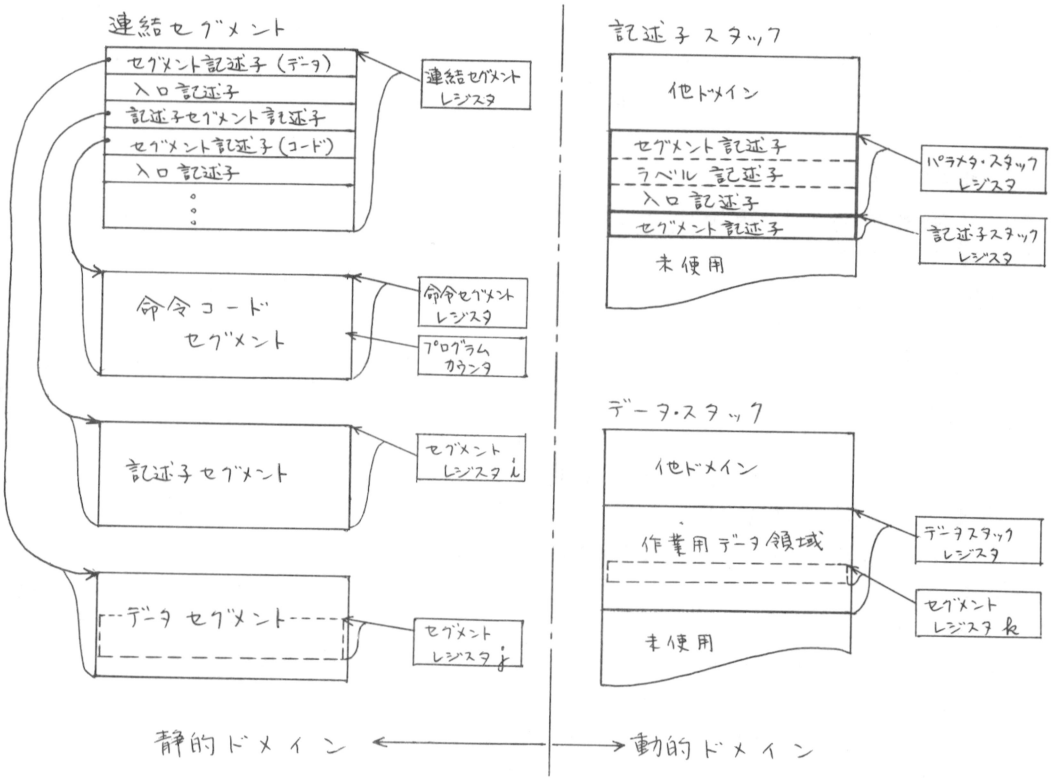
連結セグメントは静的に決定されているアクセス権の完全なリストであるので静的ドメインと同一視される。

3.2 動的ドメイン

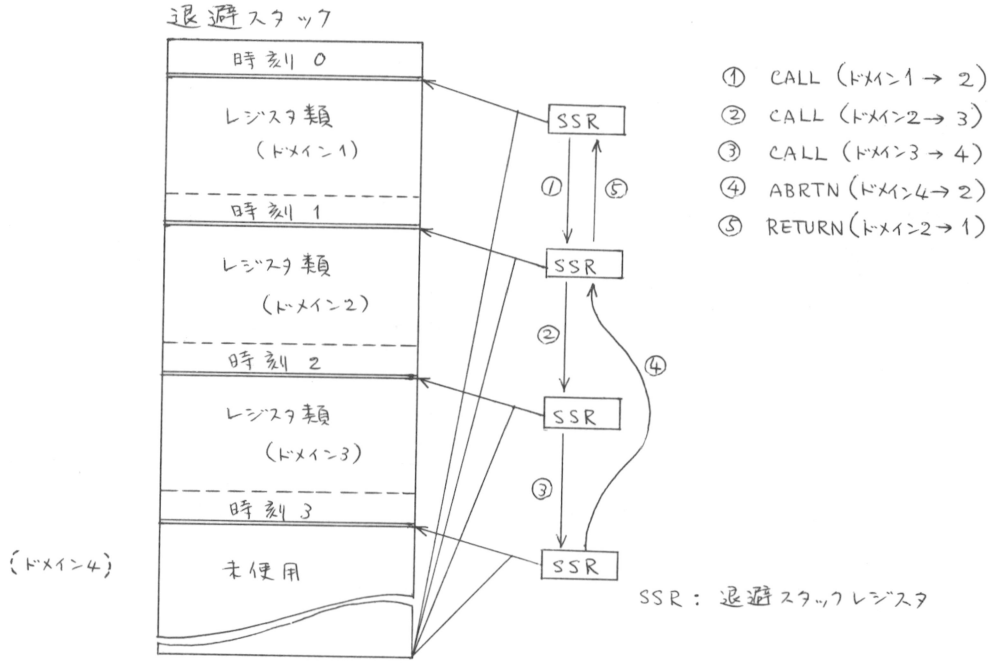
保護・許容範囲（ドメイン）は時間の関数として見たとき必らずしも一定ではなく，パラメタとして呼び出し側からアクセスを許される領域や，PL/I，ALGOLなどで用いられる動的データ領域などは，ドメインがアクティヴになるたびに変化する。これらの動的なドメインの拡張や制御のためにプロセス単位に，パターン・データ用，アドレス・データ用の二つのスタックを設ける。

A. データ・スタック
B. 記述子スタック

呼び出し側手続きからのパラメタは，そのパラメタ・データのみを示すように縮小された記述子で渡される。この記述子が積まれるのがこのスタックであり，受け取り側ドメインにはこのパラメタの個数分の記述子をまとめてフレームするパラメタ・スタック・レジスタが渡される。パラメタの記述子の参照



図－2　ドメインの構成



図－3　退避スタックと CALL/RETURN/ABRTN

は，そのレジスタの示す中でのオフセットによるので，パラメタ記述子以外をアクセスすることは不可能である。このような形でパラメタ・メカニズムのシステム的統一が実現される。

3.3 退避スタックとドメインの切替え

手続き間の呼び出し・復帰に伴うドメインの切替えの際に，呼び出し側のドメインの退避と復帰ができるように退避スタックを用意し，プロセッサ状態を示す情報の自動的プッシュダウン・ポップアップが行われるようになっている。

A. CALL/RETURN

CALL命令では，その直前のドメイン状態を示すレジスタ類（連結セグメント用レジスタ，パラメタ・スタック用レジスタ，その他のプロセッサ状態を表示するカレンダークロックなどのレジスタ等）をすべて退避スタックに退避し，次に記述子縮小指定などによりパラメタ記述子をパラメタスタックに用意し，その後各記述子型レジスタ類を，このCALL命令が指定した入口記述子が示す目的ドメインの初期状態にする。最後にプログラム・カウンタに入口記述子に示されるオフセットを設定して目的ドメインへの移行を完了する。

RETURN命令では退避スタック内の先頭フレームの値により必要なレジスタ類の回復を行うことにより呼び出し側のドメインに復帰する。

B. SVC，割り込み，割り出し

SVC（スーパバイザコール），割り込み，割り出しについてもCALL命令と同様に退避スタックへのプッシュダウンが行われるが，用いられる入口記述子は，それぞれハードウェア上の固定番地のものが用いられ，プロセッサは特権モードになる。

C. 非正規復帰とラベル記述子

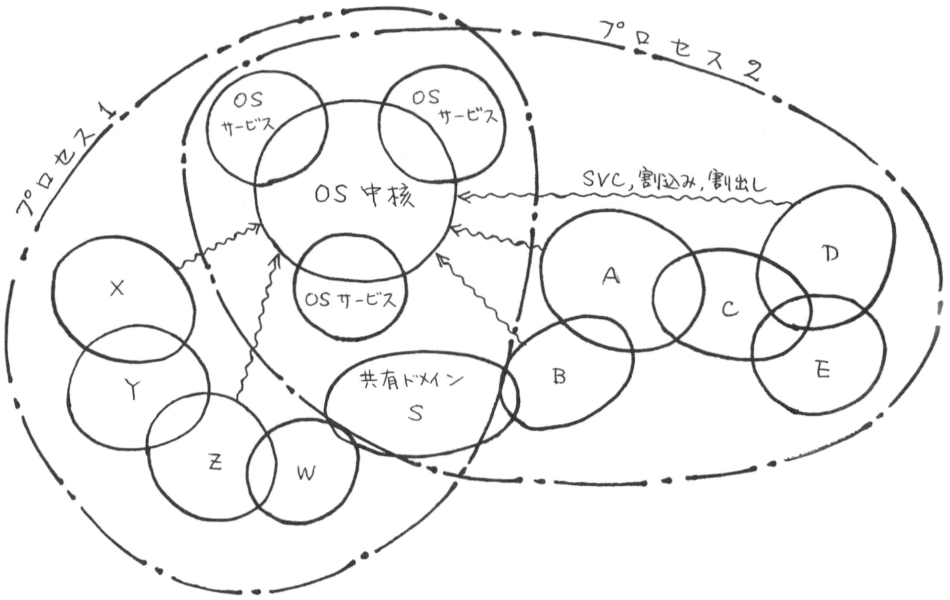
プログラムで何か異常あるいは再帰呼び出しで目的条件を検出したときなどに，単なるRETURN命令によって直前のドメインへ戻るのではなく，幾世代か前のドメインへ戻りたいことがしばしば起こる。この非正規型の戻りのために必要なのがラベル記述子であり，あらかじめ動的に作られている必要がある。この記述子を指定したABRTN命令が出されると，ハードウェアは退避スタックのトップ・フレームから順に時刻情報をラベル記述子中のそれと比較してゆき同じ時刻のものを見つけて，その直前の（時刻的に直後の）退避フレームを使ってRETURNと同様なレジスタ類の回復を行う。ただし命令セグメントおよびプログラム・カウンタにはラベル記述子に指示されたものを用いることにより目的ドメイン中の目的地点へ制御が移される。

4. ソフトウェア・システムとドメイン

図にソフトウェア・システムのドメイン構成の一例を示した。各プロセスは通常複数個のドメインを含んでおり，それらは次のような種類のものである。

A. OS中核部

オペレーティングシステムの中核部は，1章で述べたレベル1の安全性のために少なくとも1個の独立したドメインを構成している。OSの中核部はすべてのプロセスに含まれていることが必要であり，スーパバイザコール，割り込み，割り出しなどにより，いつでも，ユーザの手続き空間から制御を



図－4.　プロセスとドメイン群

受け取り，取り上げることができるようになっている。

B. OS サービス モジュール群

オペレーティング・システム中には，その中核部から制御を受け取り中核部に保存されている種々の情報をアクセスしながら，いろいろなサービスを行うモジュール群が存在する。これらの一つ一つは個々に独立した仕事を行う場合が多いので，各々が一つのドメインを構成する。このようなドメイン分けにより OS 内部での保全性が高められる。

C. ユーザ・ドメイン群

ユーザ手続きの空間には，通常複数個のドメインが見られる。1プロセス内のユーザ・ドメイン群全体としては，1章レベル2で述べた各被管理空間の分離が達成されており，各ユーザドメインによりレベル3の手続き空間の分離を達成することができる。各ユーザドメイン間では，データセグメントの共有やパラメタの受け渡しにより共通部分を持ち，これにより情報の受け渡しがなされる。

D. ユーザ共有ドメイン

ユーザ空間が，一個のプロセス内でも一様ではなく，より小さな，ドメインという単位により構成されているため，複数個のプロセス間で任意のユーザドメインを共有することが，保護機構に抵触することなく実現できる。

5. むすび

以上で，アドレス・データと通常データの分離を基礎に，動的保全機構の実現の仕方，種々のスタック，パラメタ受け渡し機構と手続き呼出しのシステム的統一がいかになされるかに言及して，ソフトウェア・ハードウェアで統一されたドメイン制御の基本構造を記述した。ここで述べられた基本的考え方は，ソフトウェアシステムの巨大化とコンピュータ・ユーティリティ指向の現在にあって，これからの進化への一方向ではないかと思う。本稿では仮想記憶化そのものにはほとんど言及しなかったが，このドメイン保護方式は仮想記憶システムとマッチして最も威力を発揮するものであり，今後も更に工夫を重ねて，残されている問題・課題の数々についても実現性を十分加味した考察と努力を継続してゆきたい。現在検討の範囲にあるテーマは，

(1) Multi-level Memory Hierarchy
(2) 多重プロセッサ制御機構での機能分散，特に(1)の制御
(3) ファイル，データベースの仮想空間化
(4) ドメイン制御機構の効果的適用によるデータベース保護方式
(5) 仮想ファイルシステムの観点からのセグメント単位のアクセス制御
(6) ドメイン制御システム下でのデータベースの同時アクセス制御

等がある。

これらの中でも特に実用性の高い項目が(6)であり，従来から提案され又実現もされている手法に見られる能率の悪さや保全性に対する不安，さらにリスタート／リカバリーの難しさ等に，この仮想記憶システム上でのドメイン保護機構が光明を投げかけてくれるであろう事を信ずる。それらに対しての努力と併行して，保護機構全体にかかるコスト，即ちハードウェア性能上の問題の解決に対しても，セグメント記述子自体の連想記憶制御や，高速スタック機構用バッファメモリの装備等の方式の具体化を考えてみたい。

なお，本稿で記述されたドメイン保護機構は ACOSシリーズ77 TOSBAC ACOS6 の仮想記憶システム上にほぼすべてが実現されている。

参考文献

1) Lampson, B.W., "Dynamic Protection Structures", Proc. AFIPS 1969 FJCC, Vol. 35
2) ACOS6 システム概説書
3) Vandebilt, D.H., "Controlled Information Sharing in a computer utility." MIT Project MAC, MAC-TR-67, 1969.
4) Apfelbaum, H., and Oppenheimer, G., "Design of Virtual Memory Systems" Proc. 1971 IEEE Internat. Comput. Soc. Conf., Boston.
5) Graham, R.M., "Protection in an information processing utility." Comm. ACM Vol 11, 5 (May 1968)