第2章

分散技術の動向

2.1　データ駆動型アーキテクチャを持つ自律分散システム

本節では，本論文で扱う自律分散システムについて述べる．自律分散システムを実現する上での基本となるのがデータ駆動型アーキテクチャである．自律分散システムには，オンライン拡張性，オンライン保守性とフォールトトレランス性の三つの特徴がある[1]．また，異種ニーズ・異種モードの共存環境下でのアシュアランス技術についてシステム構築の観点から調査し，以下にまとめた．

2.1.1　　自律分散システムのコンセプト

(1)　自律分散システムの特徴

システムの大規模化と共に，各システムがネットワークを通して広域に接続されるようになってきている．しかも社会，経済の頻繁な状況変化へ対応し，各システムもそれに適応し，変化に対応して行くことが求められるようになってきている．

このような背景の下，自律分散システムの特徴としてオンラインプロパティがある．具体的には以下の 2 つの特徴をいう．

① オンライン拡張性；

システムの周囲の状況に合わせてシステムを建設，変更，修正を行わなければならないことがある．更にこの変更，追加があっても，他のサブシステムの稼動を止めることは許されないようになってきている．このようなシステムのオンライン稼動中に部分的な変更や段階的な構築ができることをオンライン拡張性と呼ぶ．最近では経済環境が頻繁に変化し，ハードウェアやソフトウェアの技術も急速に変わるようになると，このオンライン拡張性が不可欠となる．

② オンライン保守性；

システムの一部を拡張したり，障害部分を修復したりした時には，必ず確認（テスト）しなければならない．システムを止めず，もちろんこの時にもシステムへの影響を考え，全面的な稼動停止は許されない．従

来はシステムが機能停止する夜間に保守をしていた．しかし夜間の業務までシステム化されてきており，またビジネスのグローバル化，サービス性向上によりシステム稼動時間が長くなり，機能を停止できる時間帯がほとんどなくなり，稼動中に保守をせざるを得ない状況なってきている．このように稼動中に保守ができることをオンライン保守性と呼ぶ．

③ フォールトトレランス性；

システムのハードウェア，ソフトウェアを含めた品質，信頼性を向上したとしても，システムが広域，大規模になるに従い，どこかで故障の生じてしまうことは避けられない．この時，例え部分的な障害に対しても，その障害波及を阻止でき，システム全体の稼動停止が避けられることをフォールトトレランス性と呼ぶ．

(2) コンセプト

自律分散システムコンセプトは次の観点に立ってシステムを定義している[4]．

① 観点；

自律分散システムでは，

（ａ）サブシステムを統合したものがシステムである．

サブシステムを，その目的，機能を持つものとして定義する．それらのサブシステムが統合されたものがシステムで，トータルシステムが前もって決まっている訳ではない．

（ｂ）システムには機能不稼動なものを含む．

システム内には，障害，建設途中，保守中などのサブシステムが全くないということは現実的にはない．システム内には，機能不稼動なものが全くないとの前提に立ったものが従来のシステムである．そのため，機能不稼動なものがあれば，それをシステムの異常状態とし，システムを停止するか，特別な運用に切り換えていた．しかし自律分散システムでは「異常が正常(常にどこかが異常になっていることが正常な姿である)」という考え方をとる．

② 性質；自律分散システムでは，上記２つの観点に立ち，次の２つの性質を定義できる．

（ａ）自律可制御性（autonomous controllability）；いかなるサブシステムが機能不稼動になっても残りのサブシステムは自らを制御できる．

（ｂ）自律可協調性（autonomous coordinability）；いかなるサブシステムが機能不稼動になっても残りのサブシステムは互いに協調できる．

③ 条件；２つの性質，自律可制御性，自律可協調性を満たすためには，システムは次の条件を持てば十分である．

（ａ）構造：均質

サブシステムは均質な構造である．構造的には均質であるが，もちろんそれぞれは異なったアプリケーションの機能を持って良い．構造的に均質とはシステムの入出力関係が同一構造ということで，いかなるサブシステムも接続できる．

（ｂ）機能：平等

サブシステムの機能は平等である．それぞれが自らの判断で制御，協調できることである．他のサブシステムに指令を出して動かしたり，他のサブシステムから指令を受けて動くのではない．

（ｃ）情報：局所

サブシステムは局所的な情報を用いて制御，協調できる．大局的情報によって，より評価を向上させることはできたとしても，自律的な制御と協調は，局所的情報だけでできなければならない．障害や拡張によって，常に変動するシステムにあっては，システム全体の構造や機能が変化し，いつ，どこまでが全体かを決めることはできない．そのため従来のように，大局的情報を前提とすることはできない．

(3) 適用アプリケーションシステム

当初は制御システムのネットワークにこの技術が適用され，その後鉄鋼，ＦＡ，大規模鉄道システムの制御に適用され，さらにインターネットサービスプロバイダシステム等，ネットワークアプリケーションへもその適用範囲が拡大されてきている．

2.1.2 データフィールドアーキテクチャ

(1) 構成

自律分散システムを実現する上での基本となるのがデータ駆動型アーキテクチャとしてのデータフィールドアーキテクチャである．図 2.1 に示すように分散するサブシステムを均質な構造を持つアトムと呼び，これらをネットワークで接続するコミュニケーション手段としてデータフィールドを持っている．１つのア

トムの構造はデータフィールドとアトム内のアプリケーションソフトウェアとの間で，データ駆動のエンジンとなるＡＣＰ（Autonomous Control Processor）から構成される．さらに１つのサブシステム内にある複数のアプリケーション間も同様の仕組みでコントロールするためにアトム内データフィールドを持っている[5]．

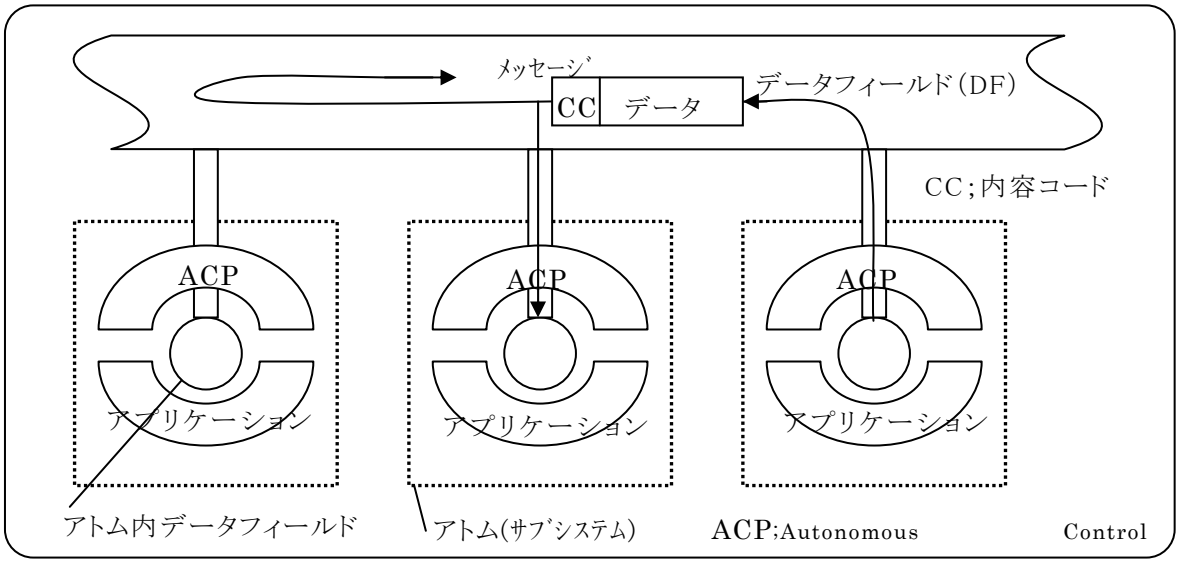


図 2.1　データフィールドアーキテクチャ

(2)　特長

自律分散システムにおいては，データは流れているもの(フロー)と見なしている．サブシステム間をながれるメッセージは内容を示す内容コード（ＣＣ：Content Code）をデータに付けて流す．これを受けて各ＡＣＰは予め登録している必要な内容コードのメッセージだけを取り出し，その内容コードに対応したアプリケーションソフトウェアを駆動する方式を採用している．このようにしてメッセージに送信アドレスを特定せず，内容を示すコードを付けてデータを流し，受信側が主体的に必要なメッセージを選択して受信してアプリケーションソフトウェアを駆動することにより，自律性を実現しているのがデータフィールドアーキテクチャの特長である．

(3)　内容コード通信方式

従来の通信は送信側が受信側のアドレスを指定していた．これに対しデータフィールドアーキテクチャでは受信アドレスではなく，データの内容に対応した内容コードを用いる．送信側はデータに内容コードをつけてデータフィールドにブロードキャストする．各サブシステムはデータフィールドに自分の必要とする内

容コードを持つデータが流れてくればそれだけを取り込む．従来は送信側が受信側を指定するもので，「選択送信」である．これに対して内容コード通信は，送信側は受信側を指定せず，受信側が自らの判断でデータをその内容で選択して受信する「選択受信」である．この内容コード通信により，いかなるサブシステムも通信に対して他のサブシステムから送信や受信を指示されることはなく，サブシステムは，その送信，受信を自ら制御でき，かつ，協調が取れるという自律性を達成できる．

データフィールドは物理的には，ネットワークであるが，メモリでも良い．1つのアトム（サブシステム）の中に複数のアプリケーションソフトウェアを同居させるのが現実解であり，このため，アトム内データフィールドを構成して，1つの計算機の中でもデータフィールドアーキテクチャを実現している．

(4) データ駆動型処理方式

従来の計算機ではソフトウェアが複数あると，それらの順番を決め，順次駆動していた．これに対してデータフィールドアーキテクチャではサブシステム内のアプリケーションソフトウェアは，必要な内容のデータがそろえば処理を開始するというデータ駆動方式である．駆動に必要なデータは複数あっても良い．複数の内容のデータに対してはデータの AND 駆動となる．OR の駆動条件でも良いが，これは入力データごとにアプリケーションソフトウェアを分けたことに対応している．処理結果はデータとしてデータフィールドに送出され，そのデータが他のアプリケーションソフトウェアに順次利用されて行く．

2.1.3 オンライン保守技術[6,59]

(1) システムニーズ

自律分散システムはオンラインプロパティ（即ち稼動中システムの段階的拡張性，システムを止めずにテストや保守を行うオンライン保守性，いかなる部分の障害に対してもシステム全体を止めないフォールトトレランス性）が特長であり，このためデータフィールドアーキテクチャにおいて，システムを止めずに，システム稼動中に，テストや保守を行う仕組みを実現する必要がある．

(2) オンラインテストの定義

テストモードのアプリケーションソフトウェアがテストデータを用いてテストするのが従来のオフラインテストである．オンラインデータをテストモードのアプリケーションソフトウェアが用いてテストしたり，テストデータを用いてオ

ンラインアプリケーションソフトウェアがテストするのをオンラインテストと呼ぶ．図 2.2 にオフラインテストとオンラインテストの定義を示す．

データ / Apl	テスト	オンライン
テスト	①オフラインテスト	②オンラインテスト
オンライン	③オンラインテスト	オンライン

オフラインテスト;①テストAplのテストデータによるオフラインテスト
オンラインテスト;②テストAplのオンラインデータを用いたオンラインテスト
;③オンライン稼動中のテストデータを用いたオンラインテスト

図 2.2 オンラインテストとオフラインテストの定義

(3) データフィールド上のオンラインテストのための仕組み

オンラインテスト実現のためにはシステム内にオンラインモードとテストモードのアプリケーションソフトウェア，及びオンラインデータとテストデータを混在できなければならない．このためデータフィールドに流れるメッセージには内容コードの他にテスト用データであることを識別するテストフラグ TF（Test Flag）を付加する．このメッセージがデータフィールドに送出されると，各サブシステムはテストフラグを基にオンラインかテストかの処理を行う．アプリケーションソフトウェアはオンラインモードと同じ環境下でテストするが，その結果として生成されたデータはテストフラグをつけてデータフィールドに送出し，それ以外の出力装置への出力を抑止する．この出力抑止により実稼動中のプロセスを妨害することはない． BIT（Built In Tester）は ACP の中の機能で，テストフラグのついたデータを受けてテスト処理するための機能である．EXT(External Tester）はテスト結果の監視や，テストデータ生成のためのアプリケーションソフトウェアである．EXT はシステム内にいくつあっても良い．EXT はデータフィールドを監視し，オンラインデータやテストデータとそれらが処理されて生成されたテストデータを対応付けることにより，テストの結果を把握できる．

オンライン保守技術により，オンライン稼動中のサブシステムを妨害することなしにテストができ，かつ現場でのテストを可能とし，テストデータ生成の負担も大幅に改善できる．

以下にオンラインデータを用いたオンラインテストとオンライン稼動中のオ

ンラインテストの２つの例について示す．

(4)　オンラインデータを用いたオンラインテスト

オンラインデータを用いたオンラインテストは図 2.3 のように実行される．テストアプリケーション（Test　APL）は，オンラインアプリケーション（Online APL）が DF からデータを受信するのと同様に，同じ内容コードを持つオンラインデータを DF から受信する．ACP はこのオンラインデータにテストフラグをオンしてアプリケーションに渡す．

ACP に組み込まれている組み込みテスタ（BIT）はテスト APL から生成される出力データに対して，テストフラグ・オンを付加する．テストフラグ・オンが付加されたデータは対応する内容コードを付加して DF に送信される．仮にテスト APL が同じサブシステムに接続されているデバイスに，あるコマンドを実行しようとすると，BIT はそのコマンドを抑制する．このことにより，テスト APL はオンラインアプリケーションの処理の実行への干渉やオンラインデータへの損傷を与えることなく，オンラインアプリケーションとテストアプリケーションの並行テストが可能となる．

また，オンラインテストによって生成されたテストデータの収集や，テスト状況の把握は外部テスタ（EXT）と呼ばれるアプリケーションソフトウェアによって実現される．EXT は DF に接続されたアプリケーションソフトウェアであり，システムには複数の EXT が存在しうる．

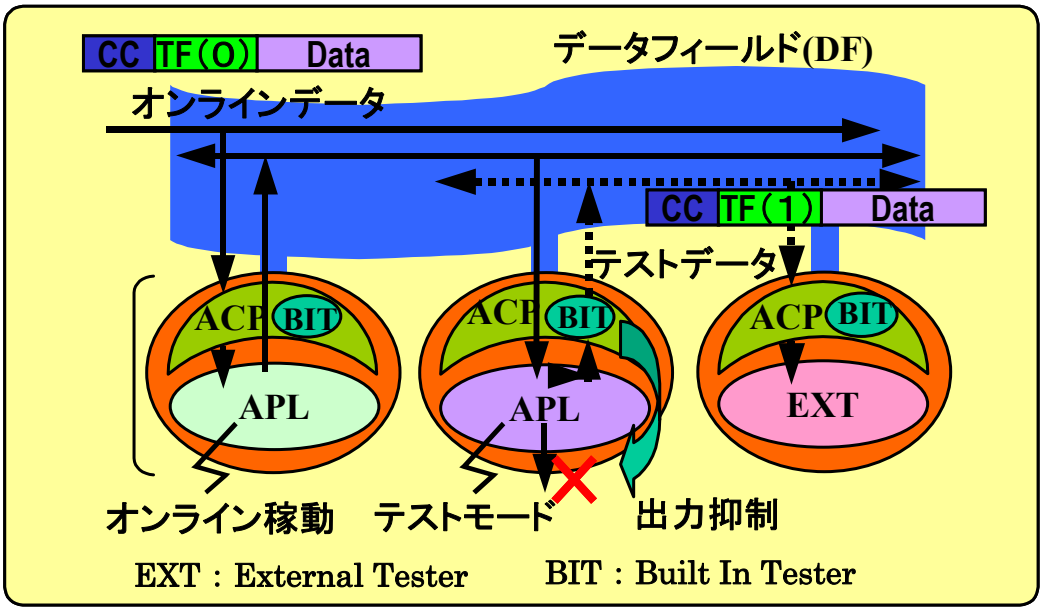


図 2.3　オンラインデータを用いたオンラインテスト

(5)　オンライン稼働中のオンラインテスト

オンライン稼働中のオンラインテストは図 2.4 のように実行される．テストフラグ・オンのデータが DF に送信されると，通常のオンラインデータと全く同様にオンライン APL によって受信され，処理される．しかし，受信側サブシステムに組み込まれている BIT は，テストフラグにより，このデータがテスト目的であることを知り，出力データにテストフラグ・オンをセットし，DF に送出する．オンライン APL からデバイスに対するコマンドもやはり BIT によって抑制される．このようにテストアプリケーションとオンラインアプリケーションは，あたかも実環境のもとでオンラインデータを処理しているかのごとく稼動することができる[24]．

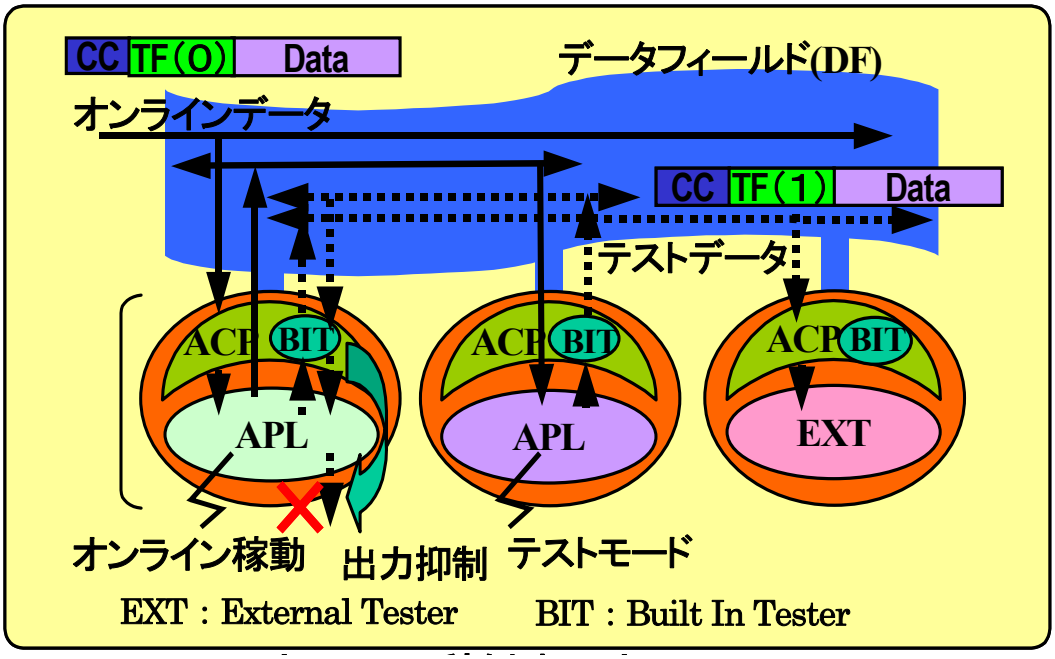


図 2.4　オンライン稼動中のオンラインテスト

2.1.4　　オンライン拡張技術[7]

システムの拡張には，幾つかのレベルがある．個々のソフトウェアモジュールや，複数のソフトウェアモジュールが格納された計算機というコンポーネントレベルから，システムとシステムとの統合というシステムレベルまである．自律構成化の考え方から，いかなるレベルも同じように対応できることも含め，それぞれのレベルでのオンライン拡張技術について述べる．

(1) コンポーネントレベル拡張技術

データフィールドアーキテクチャでは，アプリケーションソフトウェアが，データフィールドとのみインターフェースを持ち，アプリケーションソフトウェア間の直接的な関係はない．そのため，アプリケーションソフトウェアは，計算機内に新たにインストールされると，ＡＣＰ(Autonomous Control Processor)に必要な入力内容コードと，処理した結果生成されるデータの出力内容コードを登録するだけで，処理を開始できるようになる．

ＡＣＰを介したデータフィールドからの受信によって得られた入力データを格納するアトム内データフィールドは，アプリケーションソフトウェアごとに分割して管理される．データフィールドへの出力データも，同様にアプリケーションソフトウェアごとに独立に管理される．このため，アプリケーションソフトウェアの追加においても他のアプリケーションソフトウェア間でのデータ利用に関して何ら影響を与えるものでない．アプリケーションソフトウェアは内部データを持たず，処理の途中で生じたデータを保持し，次の駆動時にこのデータを処理に用いることはない．このため，アプリケーションが拡張したアプリケーションソフトウェア間の関連が変化したとしても，影響を受けずそれらソフトウェアは処理を続行できる．

複数のアプリケーションソフトウェアが格納されている計算機の拡張とは，論理的には，データフィールドにサブシステムを追加することである．そのため，いかなる他のサブシステムの変更も要せず，処理を停止する必要もない．

(2) システムレベル拡張技術

複数のシステムが互いに接続され，新たなシステムとして拡大するとき，二つの統合方式がある．第一は，各システムが構成するデータフィールドを，統合によって一つにする方式，第二は，それぞれのデータフィールドがゲートウェイを介して接続され，元々のデータフィールドを保存する方式である．データフィールドでは，そこに接続されたサブシステムがすべてのデータを共有できる．データフィールドが一つになるということは，異なったシステムのデータすべてがデータフィールドに流れ込むことである．詳細は２．３節で述べる．

2.1.5 自律分散システムの応用動向

自律分散システムは，当初は制御システムのネットワークにこの技術が適用され，その後鉄鋼プラント制御，ファクトリーオートメーション（ＦＡ），鉄道システムに適用され，さらにインターネットサービスプロバイダシステム等，ネットワークアプリケーションへとその適用が拡大されてきている．

ここでは鉄道の列車制御システムとファクトリーオートメーションシステムでの応用事例を述べる.

(1) 車上自律分散型の新しい列車制御システム（デジタルＡＴＣ）[19]

高密度運転線区の列車制御システムでは高輸送力，高安全性，高信頼性が求められる．従来は固定した区間ごとに列車の存在を検知し，そこに列車を進入させるかどうかの判断により，各列車の速度を地上の集中制御装置で求め，それを各列車の自動列車制御装置（ＡＴＣ；Automatic Train Control）に指示していた．このため下記のような問題を持っていた.

① 列車本数を増やすことができない
区間ごとにブレーキがかかったり，緩んだりすることで，無駄な時間が生じる上，列車間隔をある程度までしか縮めることが出来ない.

② 下位の速度区間に入ると急激なブレーキがかかる.
区間ごとに許容速度が決められているため，下位の速度区間に入ると急に急激なブレーキがかかり，低速になればなるほどその傾向は顕著に表れ，乗り心地が極めて悪くなる.

③ 地上装置が重厚長大である.
地上装置に速度情報，ブレーキ情報などすべての情報を持たせているため，設備が複雑となる.

この問題を解決するため車上自律型のデジタルＡＴＣ（Ｄ－ＡＴＣ）が開発された．この新しい列車制御装置は自律分散システムのコンセプトにより作られている．即ち，各列車が自らの位置検知を行い，地上からは停止する位置のみを列車に伝送し，車上ではそれに基づく列車速度の決定と制御を自律的に行う自律分散型ＡＴＣシステムである.

地上システムは分散配置としてネットワークで結ぶ構成とし，車上システムは地上からの停止点情報をもとに自律的に列車のブレーキ制御を行う自律分散システムとしている．地上システムは，ネットワーク範囲内に在線する列車の位置を把握し進路開通情報などを加味して，各列車にデジタル電文として送信する．従来のアナログ伝送方式と異なりデジタル符号伝送のＣＲＣチェックによって安全性は飛躍的に増大している．このように各論理部に自律性を持たせた構成とすることにより，システムの異常が全体に波及することを防止するとともに，段階的な構築も可能となっている.

一方，車上システムは，地上から停止する位置情報をもらうだけで，車上におい

て自身の速度と，勾配・曲線などの地理的条件を加味してブレーキ制御をする．したがって，新しい車両の機能が向上すれば，地上のシステムを変更しなくても，その車両の性能に合わせたブレーキ制御をすることができるようになり，自律性を持った車上システムとなっている．その結果，将来列車本数を増加させることがきわめて容易となり，混雑緩和となり，乗客へのサービス向上につながっている．

(2) 東京圏輸送管理システム（ATOS）

首都圏の列車輸送管理システムは制御と情報という異質なミッションであるオンラインプロパティと異種プロパティを満足するように設計されている．列車の運転を止めることなしに段階的にシステムを構築するという要請とすでに稼働しているシステムに影響を与えないようにするという要請との両方を満足している．このシステムは ATOS(Autonomous Decentralized Transport Operation Control System)と呼ばれ，10 年以上の開発期間を経て，1994 年に中央線で使用開始された[3,60]．構築に当たっては，駅システムを順次構築していって最後に線区全体のシステムが出来上がるという段階的構築手法が取り入れられている[61]．

① 駅システムの構築
② 駅ネットワークを通じて同一線区の他の駅システムとの接続
③ 上記２ステップの繰り返し
④ 駅システムが数割出来た段階で，運行モニターシステムの構築
⑤ ③ステップ
⑥ 駅ネットワークが出来てきた段階で，線区ネットワークとの接続
⑦ 線区ネットワークと輸送管理システムとの接続

駅システムは GW（ゲートウェイで結ばれた）制御と情報の２つのサブシステムよりなり，異種のデータを同時に扱えるように構成されている．

このように ATOS システムは大規模システムであり段階的に構築していき，駅数を次第に増やし，線区を構成し，さらに線区を結んでゆくというボトムアップ手法により構築されている．

自律分散技術を使ったことによって，システムは完成した駅から逐次使用開始していくというオンラインプロパティが実証された．使用開始後も駅システムの改修，機能追加，線区指令装置の増設が頻繁に行われているが，システムの稼働停止に至ることは無く，列車乱れが生じた後の平復時間（ダイヤが平常に戻るまでの時間）も大幅に短縮されるなどの効果が上がっている．1992 年 6 月にまず中央線（東京

ー甲府間）から構築が開始され，1994 年 3 月に第 1 号となる相模湖駅装置が完成し，稼動を開始した．以降，次々と駅装置と線区システムを構築し稼動を開始しながら，（2006 年 3 月末）現在は首都圏の主要線区（中央快速・緩行線，山手線，京浜東北線，横須賀線，総武本線，東海道線，常磐線（上野～羽鳥間），東北線，高崎線，埼京線，川越線，南武線など）に導入されている．

(3) タイヤ生産管理システム

近年，四輪駆動や RV 車などの SUV（Sports Utility Vehicles）など，市場での急速なエンドユーザニーズの変化や多様化により，これまで以上に迅速，且つきめ細かなタイヤの生産が要求されている．タイヤの耐久性や性能など，顧客ニーズにマッチした高品質な製品をタイムリーに提供する必要がある．

また，タイヤはさまざまな用途に使用されている．スクータのような小さいものからジャンボジェットのタイヤ，あるいは，鉱山などで使用される 1 本が数トンに及ぶ建設車両用タイヤまである．このような多種多様なタイヤの構造や製法はかなり複雑である．原材料は，天然ゴム，合成ゴム，カーボン，配合剤，補強材のナイロン，ポリエステル，スチールワイヤー，アラミド繊維など数百種類以上にのぼる．これらの原材料を加工して，つぎの工程で使用できるような中間部材を作り，成型工程，加硫工程，検査工程など 15 工程以上を経て最終的な製品であるタイヤが完成する[69]．

ブリヂストンでは，1980 年代から工場内のネットワーク化を進めてきた．当初は，制御機器メーカ独自のハードや，プロトコルを使用したネットワークであったが，80 年代後半からは，Ethernet によるネットワーク化を進め，物理的な部分は標準化をすすめていった．しかし，情報アクセス技術はまだバラバラであった．その後 90 年代になって，クライアントーサーバ型による情報活用を進めていった．

そのような経緯から，ブリヂストンは情報を加工して蓄えることなく，生のまま流す考え方に着目した．生データを，品質，性能，稼動等の情報に分類し，たとえば，長さ，重量，厚さ，製品タイヤの運動特性など 5,000 項目を超える情報に分類・整理した．現時点で使用されていない項目も多数含んでいる．この生データを「広義の改善」活動を行うための情報基本単位とする．基本単位である生データを利用者が自由自在に収集して，利用する側で必要に応じて加工して取り扱えるようにすることで，製造現場部門のみならず，製品開発・設計部門にわたる顧客ニーズに立脚した，改善活動の多様な情報要求対応できるようにした情報利用のコンセプトを作り上げ，情報の Just In Time を実現することを目指した．

このような仕組みを FOA（Flow Oriented Approach）と定義した．FOA では，工場内のネットワークを活用して，情報すなわち生データが発生のつど送出する発信

型の情報システム構造を基本にしている．そこで FOA を実現するアーキテクチャとして柔軟性，拡張性を備えていた自律分散アーキテクチャを採用することとなった[1,11]．90 年代始めには自律分散はまだメーカ独自プロトコルであったが，メーカにオープン化を促し，かつ，他社制御機器メーカにもプロトコル実装を働きかけ，ユーザ主導での標準化（ADS-net：Autonomous Decentralized System network）を進めた[70]．自律分散システムのデータフィールドアーキテクチャは，すべての情報を蓄えずに流しておき，必要なユーザが選択受信するという考え方で，まさに FOA が狙っているコンセプトと合致していた．

(4) 新聞生産工程管理システム

デジタル技術の急速な進歩により，情報伝送媒体は大きな変革期を迎え変貌してきている．インターネット上で個人組織を問わず，さまざまな情報を発信し，また，情報を収集しているなど，既存のメディアを使わない情報交換が容易に出来る状況となっている．このインターネットを媒体として使用すれば，現在のように新聞をいちいち紙に印刷し配達するという非効率的な作業がいらなくなってしまう．

21 世紀になり，このように新聞を取り巻く状況が大きく変わってきている．当然ながら，読者が新聞に求めているニーズは変化し，多様化してきている．新聞の持っている「モノと情報が一体化」しているという特徴は，他のマスメディアにはないもので，メディアとしてその価値を保ち続けるには，状況やニーズの変化を先取りできるシステムの構築が求められた．

このニーズに基づいて開発されたシステムが新聞生産工程管理システムである．新聞の生産工程は，刷版工程，印刷工程，宛名工程，積込工程の４つからなっている[2]．

刷版工程 ： 大量生産の入り口であり，新聞専用の印刷機であるオフセット輪転印刷機用の版である刷版を作成する．

印刷工程 ： 高速で低コストが求められる印刷であり，朝刊で一時間に８万部の印刷をしている．

宛名工程 ： 店単位に細分化した，宛名を新聞の束ごとに書き込む．

積込工程 ： 分単位で着発するトラックに新聞の束を仕分けして積み込む．

このように，作業内容もスピードも違う種類の作業をシステム化することは，相当の難作業になるといえる．しかも，刻々変化するニュースバリューへの対応，顧客数の変更への対応などをも同時に行う必要がある．

これらの処理能力の違い，変化への即応を可能ならしめたのが自律分散アーキテクチャである．本社システムと工場システムとを一つのデータフィールドで結び，工場内の各工程のサブシステムを別のデータフィールドで結んで，制御データとサイズの違いを吸収するようにし，また，各レイアが自律してデータフィールドを介してデータを共有することにより，フォールトトレランス性をも実現できている．

朝日新聞の世田谷工場で初めて採用された後，より多くの工場で汎用的に使用できるように修正を加えた後，1997 年に川崎工場に導入，他の工場にも次々と導入されていった．また，1998 年春には大阪と西部の両本社にもシステムが入って完成した．

2.2　アシュアランス技術

アシュアランス技術はシステムの安定稼動を保証する技術で，異種性と適応性の２つの要素を考慮したシステムに適用される技術の総称である．異種性と適応性を持ちシステムの安定稼動を保証するシステムをアシュアランスシステムと定義されている[14, 15, 16, 17]．

2.2.1　　異種性

システムは多くのニーズを満たすものとして構成されるが，このニーズの要求度合いはシステムにより異なっており，これをニーズレベルと呼ぶ．

システムの稼動の保証に求められている評価基準は様々であり，例えば以下のような異種の評価基準が挙げられる．

① 信頼性
② アベイラビリティ
③ フェールセーフ
④ 安全性（セキュリティ）
⑤ リアルタイム性

システムを設計する際にはこれらの評価基準ごとにニーズレベルを満たす必要があるが，そのレベルはまちまちである．例えば制御システムにおいては一般に信頼性とリアルタイム性が求められているのに対し，情報システムではアベイラビリティや安全性が求められる．もちろん，これらのレベルをすべてのシステムのニー

ズよりも高く設定すれば，すべてのシステムのニーズを満たすことができるが，それは現実的ではない．そこでシステムごとに要求されるレベルを満たすことになるわけであるが，システムのネットワーク化の背景により，ニーズレベルが異なるシステムが接続されていることから，システムが単独でニーズレベルを満たすことができたとしても，他のシステムと連携した時にはその限りではない．

異種のニーズレベルを持ったシステムの稼動を保証するためには，異種のニーズレベルの共存を許容するシステム技術と異種の評価基準を統合した新たな評価基準が必要となる．

2.2.2　適応性

ユーザニーズが異種性を持つのみでなく，それが時間的に変化することが，近年より顕著になってきている．

このような異種性を持つユーザニーズの変化に対応して，システムとしても適応して行かねばならない．従来のように，システムを量的に順次拡大するだけでは解決しなくなってきている．ユーザニーズの変化に合わせ，システムを質的にも適用できるような対応が必要になってきている．こういった変化するニーズに対応するためには，いかなる状況の変化が起こっても，常に柔軟に対応しうるシステムの構築が必要となる．このような状況変化への対応能力を適応性と呼ぶ．

2.2.3　アシュアランス性

時間とともに変化する異種のニーズに適応でき，システムの稼動を保証できうる性質をアシュアランス性と呼ぶ．

システムを適応させるには，システムの中に異種のモード（建設，障害，テスト，拡張など）が当然共存しうる．特に，この異種モードを共存させたとしてもシステムの稼動を保証する性質をオンライン・プロパティ（on-line property）と呼び，これを実現したのが自律分散システムである[4,5,6,7,8]．このオンラインプロパティは，３つの性質，フォールトトレランス性，オンライン拡張性（段階的拡張性），オンライン保守性(稼動中テスト/修復性)からなる．アシュアランス性実現のために，この自律分散システムのさらなる研究が進められている．

これまで，このアシュアランス，自律分散システムの技術は，一般産業，情報サービス，鉄道，宇宙などの分野を主な対象としてきた．以上のような社会インフラに対する要求が強くなるにつけ，社会インフラでもアシュアランス技術が検討されるようになってきた．

2.2.4　アシュアランス技術の動向

アシュアランス技術は，米国国防省やエネルギー省，また，日本の研究者も加わり，1996 年設立した IEEE 主催になる HASE(High Assurance Systems Engineering Symposium)[10]がきっかけとなり，研究が加速された．日本でも，電子情報通信学会の FTS 研究会のもとに第 2 種のアシュアランスシステム研究会が 2000 年秋に発足[3]，以来，活発な研究活動が進み，北米以外では初めて HASE が 2002 年 11 月に日本で開催された．米国などでは，このアシュアランス技術は，国防省ではインターネットなどの分野で，エネルギー省では，電力(原子力発電を含む)，交通（BART など)，都市公共設備など分野で検討されてきた．日本では，アシュアランスの言葉は用いられてはいなかったものの，この技術では世界をリードしており，既に実用化も進んでいる．以下，鉄道分野での実用例を紹介する．

国鉄の民営化以降，JR 各社は，鉄道輸送の安全性のみならず，乗客へのサービス向上のためのシステムを研究，開発してきた．この乗客へのサービス向上を目的とすることにより，ニーズの異種性，そして適応性，つまりアシュアランス性を満足させなければならなくなり，これを最初に実現したのが，世界最大規模の鉄道輸送管理システム　ATOS(Autonomous Decentralized Transport Operation Control System)である．このシステムは，従来のような列車追跡，進路制御といった制御システムのみでなく，乗客への情報サービス，柔軟なダイヤ計画サービスなどの情報システムを統合している．さらに，このシステムは，東京圏 19 線区，288 駅，路線長 1045.5km，1 日あたりの乗客数約 1400 万人（2005 年度版 ATOS 見学パンフレットより)を対象とする世界最大規模の社会インフラである．規模が大きいため，このシステムは，1990 年の計画以来　1992 年中央線での建設開始，1996 年実用化から，順次他線区でも開発，実用化を進め，現在でも続行中である．このように開発期間が長いことからも，ユーザニーズを完全に予測できず，システムの適応性は不可欠である．また，情報と制御システムが連携して，それぞれの異種の要求を満たし乗客指向の運行を実現し，アシュアランス性を実現した自律分散システムの技術が用いられている[20,21]．

また，乗客へのサービス向上のため，デジタル自動列車制御装置（D-ATC）も開発し，順次，新幹線や東京圏の列車に導入を進めている．このシステムは，列車自体を自律化させるもので，各列車が位置を検知し，列車相互の交信により，それぞれで走行速度やブレーキ制御により安全な運行を保証するものである．これにより，従来のような地上側での集中的な制御でなく，状況に応じた制御ができるため，列車間の間隔を狭めることができ，列車の混雑度を緩和できる．このシステムはもちろん，地上側のシステムとの連携も必要である．また，このような装置は，全車両

に同時に設置することは経済的にも難しい．そこで，順次，この装置を導入し，線路上には，新旧の装置を搭載した列車が混在する．当然，安全性は，実環境下でテストされなければならないが，営業時間の増大により，テストも，営業時間内にやらなければならないようになってきている．このような，稼動／テストモード，新／旧装置搭載列車を共存させるといった異種ニーズと，運行状況の変動に合わせた適応性からなるアシュアランス技術が研究，実用化されている[18]．

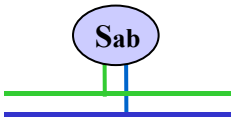
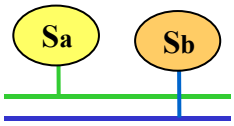
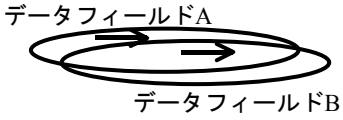
この他，アシュアランス技術は，宇宙の分野で民生品を用いた高信頼，高機能，高性能な高アシュアランスの宇宙用搭載コンピュータの実現を目指してアーキテクチャが提案されている[22,23]．また，医療分野でもペースメーカのような埋め込み型医療装置は，機械的機能，電気的機能，ソフトウェア機能は分解されてデザインされ，各サブシステムはそれぞれに対する要求を満たすものとして設計され，そのうえで再び統合されている．このように，異種の目的を持ったサブシステム群を統合し共存させることはアシュアランス技術の一つであり，今後の研究に期待される．以上のように，アシュアランス技術は情報サービスシステムなどの社会インフラ向けに，研究，実用化が進んでいる．

2.3　異種システム共存技術

自律分散システムは，システムを構成している要素であるデータフィールドとサブシステムとからできている．また，自律分散以外のシステムでもシステム同士が互いに結ばれている場合，個々の結合が１つのデータフィールドであると見なすことができる．

共存のための技術について，基本的な構成方法をデータフィールドに対して３つ，サブシステムに対して２つ提示し（表 2.1），その構成方法の組合せによって異種システム共存の６のタイプを説明する．

表 2.1　システム共存の基本的構成方法

	データフィールド	サブシステム
統合型		
分離型		
共存型		

2.3.1　　データフィールド

データフィールドの性質は，そのデータフィールドを構成している伝送速度，光ファイバー，無線，メモリなどの物理的な要素とデータフォーマット，データ形式などの論理的な要素によって次の３つに分類できる（表 2.2）.

表 2.2　データフィールドの三つのタイプ

論理的＼物理的	同じ	異なる
同じ	統合データフィールド	分離データフィールド
異なる	共存データフィールド	

(1)　統合型データフィールド

物理的にも論理的にも同一のデータフィールドを統合型と呼ぶが，以下に述べる分離型，共存型のデータフィールドを２つのシステムが共存できるようにしたものも含むものである．前者を一体型統合データフィールド，後者を接続型統合データフィールドと呼んで区別する．

システム双方が同じように相手のデータを利用し，かつ，時間的にもほぼ同時に

連携を図りたいという要求がある．また，状況に応じてアプリケーションプログラムを別システムの計算機に移行して実行することで，信頼性や応答性を改善したいという要求がある．このとき，データフィールドを統合して一つにまとめる．このデータフィールドの統合は，全く物理的にも１つのネットワークとすることによって実現できるが，２つのネットワークを接続するのみの対応でもよい．もちろん，別に１つネットワークを新たに構成することでも対応できるがコスト面で得策でない．

一体型統合データフィールドに流れるデータは，これまで別々に流れていたすべてのデータが合流するため，ネットワーク上のデータ伝送量は増大する．各サブシステムは，内容コードを基に選択受信するため，どのサブシステムからのデータも受信できる．

接続型統合データフィールドでは，全サブシステムを複数のデータフィールドと接続してデータ伝送量の増大を回避することができる．この方法によると，内容コードを基に選択受信するため，どのネットワークから受信してもよい．また，計算機は，それぞれのネットワークに同一データを重複して送信しても問題ない．このときは，受信側の計算機で，同一内容のデータを複数受信でき，ネットワークのフォールトトレランス性を向上させることができる．このように，ネットワークの負荷分散による性能向上とフォールトトレランス性向上を容易に実現できる．

(2) 分離型データフィールド

分離型データフィールドは物理的に異なる媒体を持ったデータフィールドとして定義される．流れるデータの論理的な構成は同一でもよく，また異なっていてもよい．このタイプの２つの分離型データフィールドを共存させるにはゲートウェイを必要とする．システムのデータフィールドを保存したままで，それぞれをゲートウェイで接続する．ゲートウェイは，接続するデータフィールド間にあって互いに共有したいデータのみを選択し通過させる．従来のシステムにおいては，システムをゲートウェイでつなぎ，その間のデータのやり取りを制御するために特別な装置を必要とする．

一方，自律分散システムにおいてはデータフィールドアーキテクチャによって以下のように行う．各計算機の ACP が，そのサブシステムで必要とする内容のデータのみを取り込むのと同じ機能である．データフィールドを構成するシステムにとっては，そこに接続されるいかなるサブシステムも自律したサブシステムと見なせる．ゲートウェイでも，それがデータフィールドに接続されれば，そのデータフィールドを構成するシステムの自律したサブシステムになり得る．つまり，ゲートウ

ェイとシステムを合わせて他方のシステムにとっての自律したサブシステムと見なせる．ゲートウェイを自律させるために ACP を持たせる．システム内には，幾つかのサブシステムが存在し，それらには，複数のアプリケーションソフトウェアがインストールされている．ゲートウェイの ACP は，これらアプリケーションソフトウェアが必要とするデータの内容コードを登録する．これにより，一方のデータフィールドを流れるデータの中から，他方のシステムが必要とする内容コードのデータのみがゲートウェイを介して取り込まれ，他方のデータフィールドに送出される．このゲートウェイから取り込まれたデータは，システム内で発生したデータと全く同様に，そのデータフィールド上を流れ，それを必要とするサブシステムで選択収集される．このデータが，他のシステムから送出されたかどうかにかかわりなく，アプリケーションソフトウェアは，それを用いて処理を実行する．また，この処理結果は，データフィールドに送出され，もし，他方のシステムがそれを必要とし，ゲートウェイにその内容コードが登録されていれば，そこを通過して他方のデータフィールドに送出される．

(3) 共存型データフィールド

物理的には同じデータフィールドの中に，論理的に別の体系のデータを流して，システム間で区別して使用する場合，これを共存型データフィールドという．データフィールドは別に存在するとみなすが，コンテンツコードが異なるデータが流れていると見れば同一のデータフィールドともみなせ，その意味では統合型データフィールドと同じ働きをする．システムのデータフィールドは保存したままであるが，これを実際に共存システムとして働かせるには，２つ以上のデータフィールドに１つのサブシステムから接続できるように構成して，サブシステムからそれぞれのデータフィールドに流れるデータを取り込めるようにするか，ゲートウェイで２つのデータフィールドを接続してもよい．

元々あるネットワークに，別サブシステムを接続し，それぞれのシステムのネットワークを用いるため，サブシステムはネットワークの本数だけ通信ポートを用意する必要がある．サブシステムの ACP は，アプリケーションソフトウェアがインストールされたときに，必要とする入力データの内容コードを登録する．共存型にするために，サブシステムをそれぞれのデータフィールドに接続する方法はいろいろ考えられるが，たとえば実際にケーブルでつないだり，無線で構成されたデータフィールドであればそれらを受信発信できるようにしたりするなどである．

この接続法では，ネットワークが複数あり，データフィールドに流れるデータは，それぞれのデータフィールドを流れるため，ネットワーク上のデータは共存させる

前と変化はしない．

2.3.2　　サブシステム

データフィールドの場合と同様，サブシステムを共存する方法にも２つの種類がある．サブシステムが２つのシステムと結ばれているか，あるいは２つの別システムのデータフィールドと結ばれているとき，いずれか１つのシステムあるいはデータフィールドとの間でしかデータのやり取りができない分離システムと，サブシステムが２つのシステムとデータをやり取りすることができる統合システムとの２つがある．以下にこの２つのシステムについて説明する．

(1)　統合型サブシステム

このサブシステムは，複数の種類のアプリケーションを持っており，これがサブシステム内に統合した形で存在する．それぞれのアプリケーションは，対応するシステムあるいはデータフィールドとの間でデータ交換を行い，その結果をサブシステム内の別のアプリケーションで処理を行うことによってシステムの共存を可能としている．

(2)　分離型サブシステム

このサブシステムには，１種類のアプリケーションプログラムしか組み込まれておらず，このアプリケーションプログラムは１つのシステムあるいはデータフィールドとしかデータのやり取りができない．したがって，システム共存を行うためにはデータフィールド側で対応を取る必要がある．

2.3.3　　異種システム共存技術

サブシステムとデータフィールドから成る２つのシステム A，B があるとき，この２つのシステムを共存させるには，サブシステムとデータフィールドの型の組合せにより，表 2.3 に示すような６つのタイプのシステム共存タイプがある．以下にそれぞれのタイプの特徴を説明する．

表 2.3 サブシステムとデータフィールドの分類

データフィールド / サブシステム	統合型	分離型	共存型
統合型	Type-1	Type-2	Type-3
分離型	Type-4	Type-5	Type-6

Type-1; 統合型サブシステム－統合型データフィールド（図 2.5 (a)）

サブシステムとデータフィールドが共に統合型であり，サブシステムからブロードキャストされるデータは統合型データフィールドに流れる．この場合データフィールドが一体型統合データフィールドである場合は，このシステムは１つのシステムとなる．一方，接続型データフィールドである場合は２つのシステムが共存できたことになる．当然のことながらデータフィールドは統合されているため，それぞれのシステムのデータは同一データフィールド内に同時に流される．このためデータフィールドの伝送容量の検討が必要となる．

Type-2; 統合型サブシステム－分離型データフィールド（図 2.5 (b)）

サブシステムは統合型で，２つの分離型データフィールドの双方に接続されている．データはサブシステムのアプリケーションプログラムが作成し，対応するデータフィールドにブロードキャストする．データの受信はデータフィールドに流れているデータを選択受信して，対応したアプリケーションプログラムがそのデータにより処理を行う．これは，統合システムによりシステムを結合することとなる．

Type-3; 統合型サブシステム－共存型データフィールド（図 2.5 (c)）

データフィールドは共存型であるため，統合型サブシステムのアプリケーションプログラムはブロードキャストするが，データフィールドに合わせた形でデータフォーマット作成する．接続しているデータフィールドからはデータを受信して処理を行う．データフィールドを介してシステムの共存が可能となる．

Type-4; 分離型サブシステム－統合型データフィールド（図 2.5 (d)）

サブシステムは分離型であるので，それぞれに別のアプリケーションプログラムを持っているが，データフィールドにブロードキャストされるデータは同一の仕様のデータとして扱われる．統合型のデータフィールドに流れているデータはそれぞ

れのサブシステムにおいて選択受信される．データフィールドの統合によりシステムは共存している．

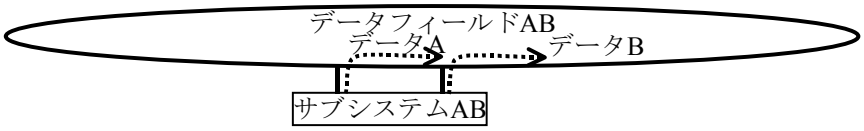
Type-5;　分離型サブシステム－分離型データフィールド（図 2.5 (e)）

サブシステムとデータフィールドが共に分離型であるということは，それぞれのシステム自体は互いに独立しているということである．これは，２つの独立したシステムである．ゲートウェイで両方のデータフィールドを結ぶことによりシステム共存ができる．

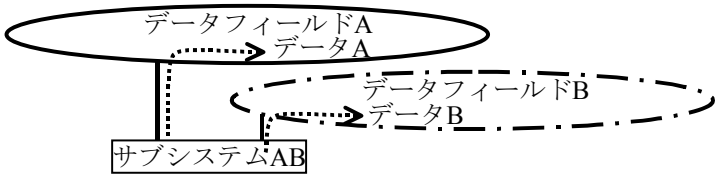
Type-6;　分離型サブシステム－共存型データフィールド（図 2.5 (f)）

サブシステムは分離型で，それぞれのアプリケーションプログラムはブロードキャストされるデータフォーマットをデータフィールドに合わせた形で作成できる．接続しているデータフィールドからはデータを受信して処理を行う．このままではシステムの共存はできていないが，センターシステム同士をゲートウェイで結ぶか，データフィールド間をゲートウェイで結ぶことによりシステム共存が可能となる．

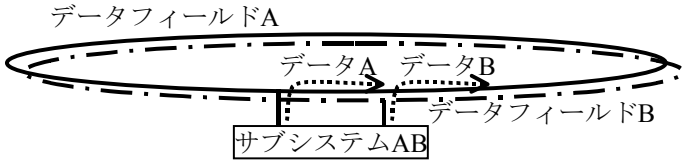
以上説明したことを整理すると，**Type-1** と **Type-4** はシステムは統合型データフィールドによってすでに共存しているといえ，**Type-2** と **Type-3** は統合型サブシステムによってシステム共存が実現でき，**Type-5** と **Type-6** はゲートウェイでデータフィールドを接続することによるシステム共存の方法である．



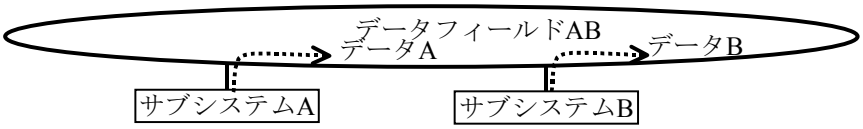
(a) Type-1　統合型サブシステムー統合型データフィールド



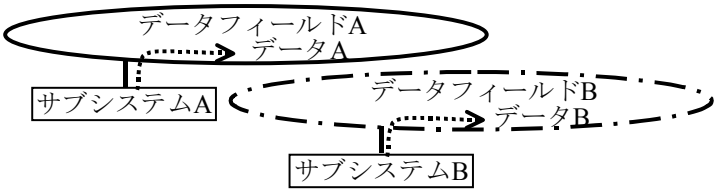
(b) Type-2　統合型サブシステムー分離型データフィールド



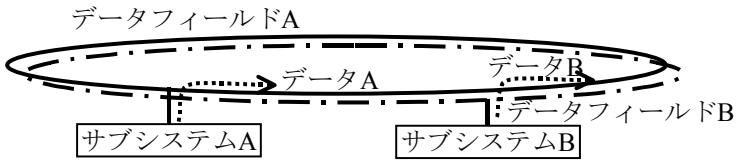
(c) Type-3　統合型サブシステムー共存型データフィールド



(d) Type-4　分離型サブシステムー統合型データフィールド



(e) Type-5　分離型サブシステムー分離型データフィールド



(f) Type-6　分離型サブシステムー共存型データフィールド

図 2.5　サブシステムとデータフィールドの分類

また，この共存型データフィールドによる共存技術を使って，異種システムの共存技術として次の２方式がある[24].

さらに，共存型データフィールドによる共存技術を使って，異種システムの共存技術としてこの２種類を統合した方式を新たに提案する.

(1) データ交換型システム共存技術

２つのデータフィールドをゲートウェイ(GW)により結び，２つのシステムの間でのデータを交換することによって，ニーズレベルの違う２つのシステムの共存，あるいはニーズの異なる他システムとの共存が可能となる．すなわち，別々のデータフィールド DF（Data Field）を有した，ニーズレベルの異なるまたはニーズそのものが異なる２つのシステムを共存させるのがこの方式である．このように，ニーズレベルもしくはニーズそのものの異なるシステム間に，相互のデータ交換処理を行う GW によってニーズの差を吸収させ，システムの共存を可能とした(図 2.6).

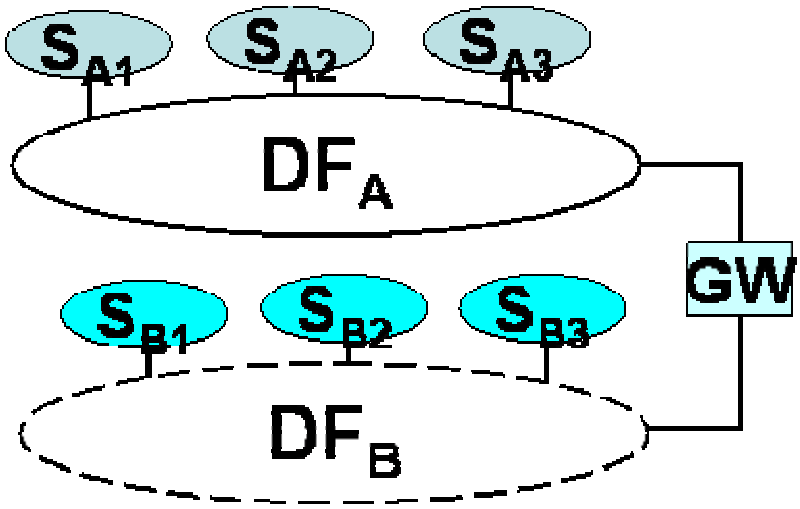


図 2.6　データ交換型システム共存技術

(2) データフィールド型システム共存技術

統合型のサブシステムが両方のデータフィールドにアクセスできるように構成する．このサブシステムは２つのシステムのデータを読むことによって，そのコンテンツコードに従ったリアルタイムな自律制御が可能となる．データにどちらのデータフィールドのデータかを区別できるフラグが付いているため，サブシステムはどちらかの処理をデータによって制御できる（図 2.7).

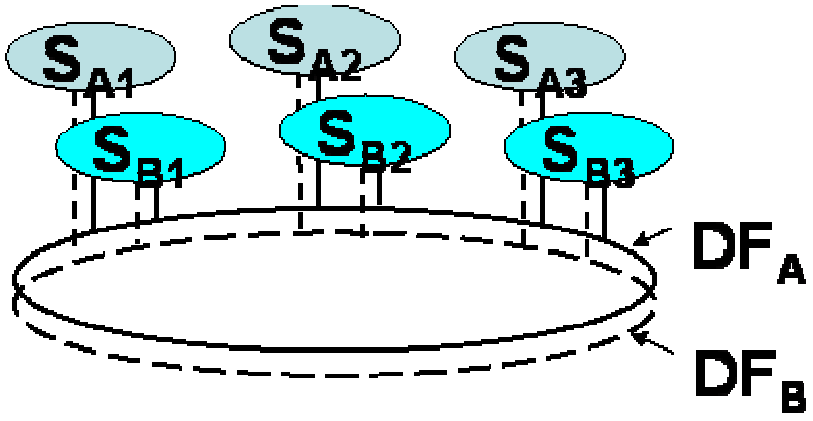


図 2.7　データフィールド型システム共存技術

(3)　データ交換・データフィールド統合型システム共存技術

データ交換型とデータフィールド型の２つの方式を統合したシステム共存技術である．データ交換型の持つ，「ニーズレベルの異なるまたはニーズそのものが異なる２つのシステムを共存させる技術」とデータフィールド型の持つ「２つのシステムのデータを読むことによって，そのコンテンツコードに従ったリアルタイムな自律制御が可能となる技術」の両方式の利点を活かしたシステム共存技術を提案する．（図 2.8）これを「データ交換・データフィールド統合型システム共存技術」と呼ぶ．

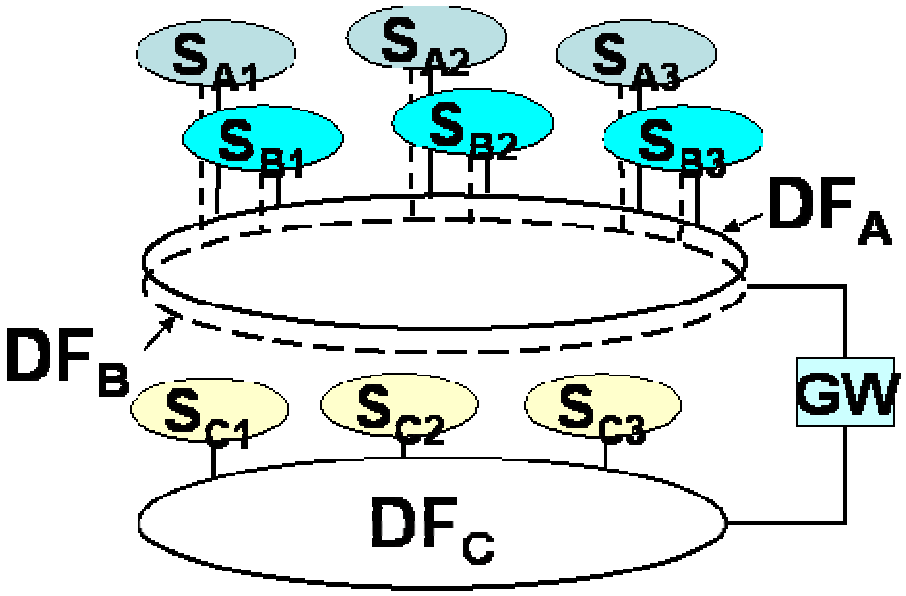


図 2.8　データ交換・データフィールド統合型システム共存技術