大規模開発プロジェクトにおける技術支援の課題

Issues on Technical Support for Large-scale Development Project

菅　原　　伸

要　約　大規模な開発プロジェクトで経験の浅い開発ツールを採用することには大きなリスクを伴う．リスクを回避し，プロジェクトを成功に導くためには，その開発ツールを熟知した専門家による技術支援が不可欠である．そのようなプロジェクトでは，技術支援の良し悪しがプロジェクトの成否に直結することがあり，どのような技術支援を提供するかは極めて重要な問題である．最近，開発・実行環境ツールとして初めて Forte を採用した大規模開発プロジェクトに技術支援担当として参加した．本稿では，技術支援を担当するにあたって何を課題と考え，実際にどのような支援を提供し，プロジェクトがどのような問題に遭遇したかについて報告する．また，プロジェクトの結果を踏まえて，掲げた課題についての評価を試みる．

Abstract　It may involve many of risks in a large-scale system development project to adopt a development tool with which the members have not well experienced. In order to avoid such risks and to lead the successful project, technical support of professionals with full experience of the tool is indispensable. Because the success of such project sometimes depends on the technical support, it is important what is provided in the support. The author recently took part in a large-scale development project in chage of technical support. The project adopted Forte, a deveropment and production environment tool, as their frist experience. In this paper reported are what had been considered as issues in technical support at the beginning, what kind of support was actually provided, and what problems were encountered in the project. Finally, the evaluation of each issue taking account of the result of the project is mentioned.

1. はじめに

開発ツール製品の足は速く，新しい製品が次々と登場し，その多くは数年で市場から消えていくことが繰り返されている．このような状況では，多くの開発プロジェクトが否応無しに経験の乏しいツールの使用を余儀なくされる．新しいツールは，便利な機能や高い生産性といったメリットをもたらす反面，新たな問題を持ち込む可能性もある．未経験のツールそのものや，その周辺に潜在する問題の予測は困難であり，予測不能であること自体が大きなリスクになる．また，潜んでいる問題の性質によっては，プロジェクトが致命的なダメージを蒙る場合もある．しかし，リスクを避けるために，いつまでも旧式ツールにこだわっていたのでは，競争に立ち遅れてしまう．経験の浅いツールを採用しつつプロジェクトを成功に導くには，ツールの特性を熟知した専門家の技術支援が不可欠であり，技術支援担当がプロジェクトに対してどのような支援を提供するかによって，プロジェクトの成否が分かれることすらあり得る．

A 社では，従来メインフレームと UNIX ワークステーションで運用していた通信サービス業務全般を支援する基幹システムを，大型 UNIX サーバと PC をベースとする分散オブジェクトシステムにダウンサイジングした．新システムの開発では，開発

ツールおよび実行環境ツールとしてオブジェクト指向分散システム開発・実行環境ツール Forte を全面的に採用した．開発期間は 1997 年 7 月から平成 1998 年 12 月までの 1 年半で，当初計画通り 1998 年 12 月からサービスを開始した．開発は A 社のグループ企業である B 社が請負い，当社他 3 社が開発に協力した．

当社は開発作業の他にプロジェクトに対する技術支援も担当し，筆者は技術支援チームのリーダを担当した．B 社と協力企業 3 社の開発チームは，少数の技術支援メンバを除いて，ほとんど全員が Forte の未経験者であった．技術支援チームの役割は，プロジェクトに対する様々な技術支援を通じて，ツールやその周辺に潜むリスクを回避し，プロジェクトを成功に導くことであった．

以下の章では，最初にプロジェクト概要を紹介後，技術支援チームとして事前に何を課題と考え，課題をクリアするために何を実施したかを述べる．更に，実際のプロジェクトでどのような問題が発生し，それらに対してどのように対応したかを述べ，最後に，プロジェクトの実績を踏まえて，個々の課題に対する評価を述べる．

2. プロジェクトの概要

開発したシステムとプロジェクトの概要，および主要な実績データを簡単に紹介する．

2.1 システム概要

新システムは，A 社の通信サービス業務全般を支援する基幹システムで，表 1 に示す五つのサブシステムで構成される．これらのサブシステムのすべてを Forte で構築した．

表 1　サブシステム一覧

ｻﾌﾞｼｽﾃﾑ名	機能概要
受注管理	通信サービスの新規契約や変更、廃止の注文の登録、変更などをおこなう
開通管理	注文に基づく設備の設計、開通・試験・課金のための設備制御、注文の進捗管理等をおこなう
設備管理	通信設備の各装置とその構成の管理をおこなう
設備制御	通信設備に対する 1 次制御インタフェース
故障管理	設備故障申告の受付，故障の対応状況の管理をおこなう
その他	システム基盤、共通機能など

2.2 システム構成

図 1 に本番環境のシステム概念図を示す．本番環境の特徴を以下に列挙する．

1） 各支店に設置した全てのクライアントは，平時は関西センタに集約したサーバ群をアクセスし，地震等の広域災害で関西センタが使用不能になった場合は，関東センタのサーバがサービスを肩代わりする．

2） 関東センタは，平時はバッチ業務，情報活用業務のサーバとして使用する．

3） 関西センタは，データベースサーバとアプリケーションサーバ（設備/故障サーバ，受注管理サーバ）を分離する形態とした．

4） データベースサーバは 2 台のパラレル構成とし，1 台のディスクアレイ装置を共用する．また，データベースサーバは，Oracle サーバ専用機とし，Forte 実行

環境および，Forte 業務アプリケーションは搭載しない．

5) Forte 実行環境および，Forte 業務アプリケーションは，関東，関西両センタの 7 台のアプリケーションサーバと，各支店等に配置したクライアント PC に搭載．

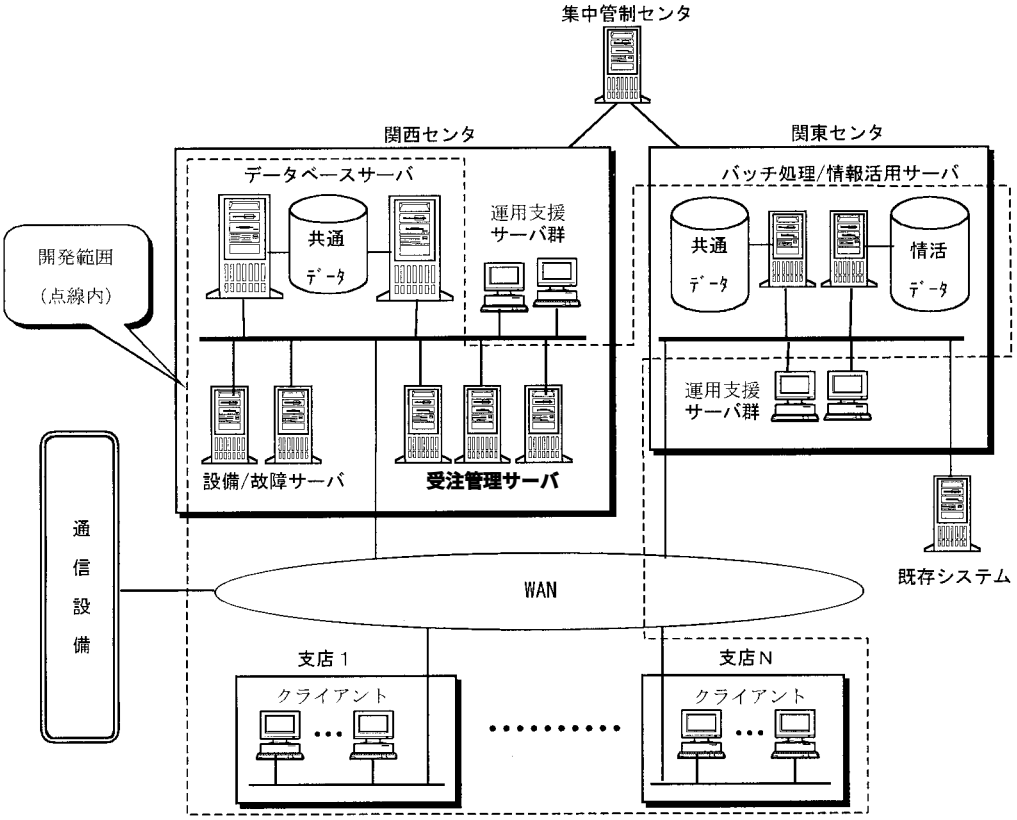

図 1　本番環境のシステム概念図

なお，開発・試験環境は，本番環境とは別に，関東，関西の両拠点にそれぞれ複数台のサーバ（UNIX と WindowsNT 混在）と Windows 95 PC を設置し構築した．

2.3 開 発 体 制

図 2 にプロジェクトの体制図を示す．体制の特徴を以下に列挙する．

1) 開発拠点が関東と関西に分かれた地域分散開発．

2) プロジェクト発足当初，Forte 経験者は技術支援チームの数名のメンバのみで，他のほとんど全員は Forte の初心者．

3) 開発の主力は，当社，X 社，Y 社の 3 社による分業体制．

4) プロジェクトメンバ総数は最繁時で約 350 名．

5) データベース設計は B 社の DB 設計チームが単独で実施．

6) 基盤チームは業務アプリケーションの共通基盤ソフトウェア開発を担当し，インフラチームは，それ以外の共通的問題全般を担当．

7) 環境チームは，結合試験環境と本番環境の設計・構築・運用を担当．

8） 技術支援チームは，関東と関西の両方に対する技術支援を担当．

9） Forte の輸入代理店（以下，代理店）のコンサルタント 1 名が非常勤（週 1 日）で技術支援チームに合流．また，Forte 社コンサルタント 1 名が，途中 1 ヶ月間，技術支援チームに参加。

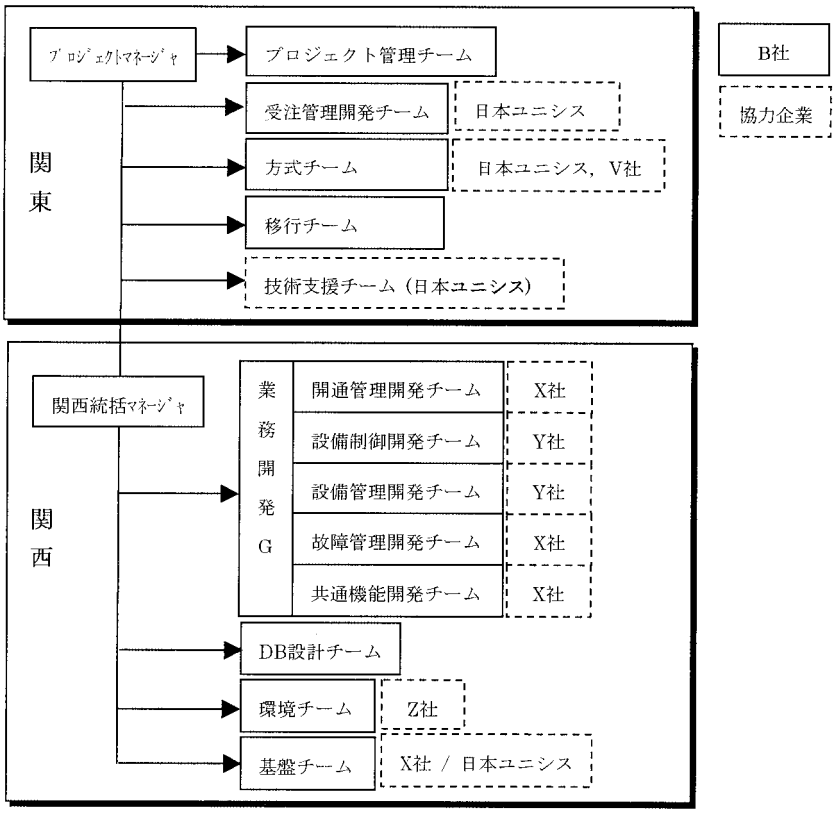

図 2 プロジェクト体制

2.4 開発スケジュール

図 3 にプロジェクトの計画と実績の線表を示す．スケジュールは，ワンパスのウォータフォール工程として計画された．実績では，物理設計および製造の着手が約 1 ヶ月遅延した．原因は，A 社が，旧システムの仕様と異なる仕様を要求したため，その調整に手間取ったこと，などであった．

製造工程の終わりでは計画線表を回復した．回復できた要因としては，大量人員投入による作業の並列化がある．また，大量の開発要員が Forte をスムースに習得できたこと，分散システム全体を容易に開発し試験できる Forte の高生産性の貢献も大きかった．

なお，スケジュールの中で，二次請負である 3 社が責任を負う作業は，外部設計から結合試験までであり，製品試験と総合試験は，一次請負である B 社および開発依頼主である A 社の責任でおこなう作業である．

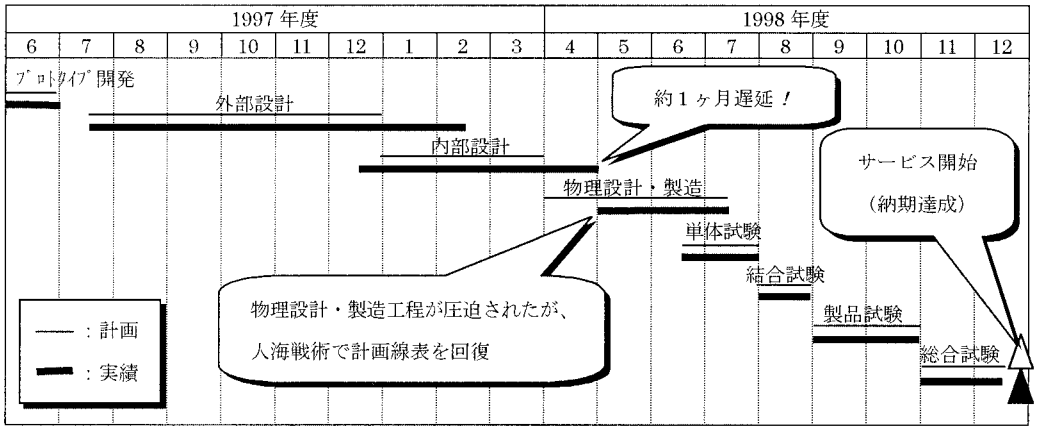

図 3　計画線表と実績

2.5　開発規模，開発工数

1)　開発規模

開発したシステム全体の Forte ソース行数は空白行，コメント行も含めて約 250 万行であった．C 言語と Forte では，1 ファンクションポイント当りのコード量は約 7 倍の差があることが知られている[1]．この倍率で換算すると，Forte の 250 万行は C 言語の約 1,750 万行に相当する．

2)　開発工数

開発に要した総工数は，B 社主導で実施された製品試験と総合試験を除いて，約 2,500 人月であった．この内，Forte に強く依存する製造工程および単体試験の割合は約 25%，600 人月強であった．

2.6　試　　験

単体試験は，主に各開発者の PC 上で，スタンドアロン実行形態で実施した．結合試験では，開発用サーバを併用した分散実行形態で実施した．製品試験では，開発用のサーバと PC で本番環境をシミュレートした擬似本番環境上で試験を実施した．この時，Forte のシミュレーティド環境機能[*1]を利用した．総合試験では，本物の本番環境で試験を実施した．

単体試験，結合試験，製品試験では，多くのバグが検出されたが，その大半は設計ミスやプログラムミスによるもので，Forte そのもののバグによる動作不良の報告は皆無であった．

2.7　品　　質

1)　サービス開始後の障害発生状況

サービス開始後にシステムダウンを含む障害が何度か発生した．ただし，その殆どはサーバのハードウェア故障や，Oracle のバグなどが原因であり，Forte 実行環境やアプリケーションの不具合が直接の契機となった障害は報告されていない．最も頻度の多かったハードウェア障害は，その後の調査で，コネクタの接触不良が原因と判明した．

2)　レスポンスタイム

製品試験で，プロジェクト計画書で規定した代表的機能の目標レスポンスタイムが，合格範囲にあることが確認された．ただ，規定外の一部の機能で，レスポンスタイムが異常に悪いことが明かになったが，その後の諸対策の実施により，サービス開始時には妥当なレベルまで改善された．サービス開始後は，利用者からレスポンスタイムに関するクレームは上がっていない．

3. Forte の概要

Forte は大規模な分散トランザクションシステムの開発から運用までカバーするALL-IN-ONE タイプの製品である．Forte を一言で言うと，通常の分散システムであれば，クライアントとサーバのプラットフォーム（ハードウェア＋OS）とデータベース製品（Oracle 等），それに Forte さえあれば，大抵のシステムは構築・運用できる製品である．Forte を利用すると，TP（Transaction Processing）モニタ製品やORB（Object Request Broker）製品などのミドルウェア，プラットフォーム毎に異なる製品等の組み合わせは不要になり，オープン製品で起こり勝ちな組み合わせ問題を回避することができる．

Forte の製品ポリシーは，大規模分散システムの開発・展開・運用に関わる様々な複雑さを可能な限り隠蔽することにある．隠蔽可能なコードや操作を Forte が代行することによって，開発や出荷作業，更に運用に関わる様々な作業が，シンプル且つ容易になり，一連の作業の生産性を上げることができる．Forte が隠蔽しているものは多岐にわたるが，その代表的なものの一つに，分散環境における通信の隠蔽がある．通信に関わる一切の処理を隠蔽することにより，プログラマは通信を一切気にすることなく，分散アプリケーションの全体を，あたかも 1 台の PC 上で動作するスタンドアロン・アプリケーションであるかのように開発することができる．複雑さの隠蔽は，すべてが順調に進めば極めて大きなメリットをもたらすが，反面，ひとたび問題が発生すると，問題の原因も隠蔽されることで，解決が困難になるというデメリットも併せ持っている．

また，Forte は専用の開発リポジトリを持っており，大勢の開発者が大規模なシステムの開発を，リポジトリを介して効率良く共同作業することができる．

Forte の詳細については参考資料[2]を参照されたい．

4. 課題認識と具体的施策

プロジェクト参加に当って，プロジェクトの特徴を次のように認識した．

① 1 次引受元である B 社を含め，開発担当各社に Forte 経験者が殆どいない．
② B 社も含め，オブジェクト指向開発方法論に馴染みが薄い．
③ B 社を含め，主力 4 社によるマルチベンダ体制である．
④ 開発拠点が関東と関西に分かれた拠点分散開発である．
⑤ 開発メンバの総勢ががかなりの数に昇ると予想される．
⑥ 計 7 台のサーバと数百台のクライアントで構成される広域分散システムである．

この認識に基づき，技術支援チームとして対応すべき課題は次の 5 点であると考え

た.

課題 1. Forte とオブジェクト指向方法論に馴染みの薄い大勢の開発メンバを如何に素早く立ち上げるか.

課題 2. 社風の異なる複数の企業に対して，如何にして公平な技術支援を提供するか.

課題 3. 関東，関西の物理的な距離の隔たりを如何にして克服するか.

課題 4. 技術支援チーム自身にも経験の無い，Forte による大規模分散システムの開発を如何にして指導し成功に導くか.

課題 5. メンバが僅か数名の技術支援チームで，大規模プロジェクトに対する技術支援を如何に効率良くおこなうか.

これらの課題を達成するため，以下のような具対策を実施した.

4.1 教　　育

Forte とオブジェクト指向方法論の知識・経験不足は前もって予想されたことであり，プロジェクト発足の半年ほど前から独自の教育コース開発を進めていた．プロジェクト参加メンバの全員とまではいかなかったが，各社の約 70 名に対して以下の教育を実施した.

① オブジェクト指向分析・設計入門（1 日コース）
② オブジェクト指向分析・設計演習（4 日コース）
③ Forte 入門（1 日コース）
④ Forte プログラミング演習（3 日コース）

各コースのレベルはビギナー向けの内容にとどまっており，この教育だけで直ちに実践に移るのは無理があるのは承知していたが，これが限界であった.

4.2 プロトタイピングと技術説明会

技術支援チーム自身の経験不足と，参加各社に対する不充分な教育を補完するために，プロジェクト発足直前の 1 ヶ月間でプロトタイピングを実施した．プロトタイプは，旧システム用の営業マニュアルのみを頼りに設計した．プロトタイピングで開発した機能は，新システムの全体からみればごく一部の機能に過ぎないが，その中には，本開発でも役立つと思われる，次のような技術あるいはノウハウを埋め込んだ.

① GUI からデータベースに至る，完結した一連の処理ルートの実装例
② 3 層アーキテクチャ[*2] の実装例
③ 有限の大きさの画面上で大量のデータを操作性良く表示するための設計手法として，独自に考案した色付きタブフォルダ・デザインパターンの実装例
④ Forte 固有の機能であるサービスオブジェクトの利用法と実装例
⑤ インタフェースと実装を分離する Facade デザインパターン[3]の実装例
⑥ データベースアクセスのボトルネックを解消するために，Forte 社が考案した DBEntityMgr デザインパターン[4]の実装例
⑦ 大量の定数を効率良く利用するために，独自に考案した 3 段階の定数キャッシング・デザインパターンの実装例
⑧ 障害対策の実装例

これらの技術やノウハウの解説資料を作成し，プロジェクト発足直後に各社に対し

て説明会を開催し，同時にプロトタイプのデモとソースコードの開示もおこなった．資料では以下のテーマについて解説した．

① プロトタイプの概要
② 3 層アーキテクチャに基づいた設計方針
③ オブジェクトの種類と設計方針
④ DBEntityMgr デザインパターン
⑤ Facade デザインパターン
⑥ WindowSwitcher デザインパターン
⑦ TabFolder デザインパターン
⑧ 定数管理と供給方法
⑨ ログの採取と制御の手法
⑩ 障害の検出と対処方法
⑪ Rational Rose による共同オブジェクト指向分析・設計の進め方
⑫ 日本語オブジェクトモデルから英語オブジェクトモデルへの変換方法
⑬ Rational Rose と Forte 間のフォワード/リバースエンジニアリング
⑭ Forte における共同開発の進め方
⑮ Forte におけるデバッグと試験

この技術移転は，未経験ツールでの開発に対して各社が抱いていた不安を緩和し，少なくともプロトタイプのソースコードを真似すれば動くものが作れる，という安心感を与える効果があった．また，技術支援チームのメンバにとっても，それまで学習し蓄積していたノウハウや考え方が間違っていなかったことを証明でき，その後の支援活動に対する自信がついたことでも意義深いものであった．

4.3 プロジェクトマネジャに対する情報提供

プロジェクト発足の前後で，B 社プロジェクトマネジャの判断を支援する目的で，以下の情報を提供した．

① 開発工程計画の考え方と各工程での作業内容の案
② 大規模分散システムを複数チームで並行して開発するための，機能分割の考え方
③ 分析・設計工程の途中経過の評価，問題点指摘，改善策の提案
④ Forte アプリケーションの応答時間の見積り手法

4.4 共通規約類策定

コーディング規約，GUI 規約，命名規約などの共通規約を自ら策定，あるいは策定を支援した．

4.5 関西に対する出張技術支援

プロジェクト発足当初は，関西側設計スタッフも関東の拠点に集結し作業していたが，外部設計工程の終盤から，関西側スタッフは関西に戻ることになった．このとき，関西側責任者から技術支援要員として 3 名の常駐支援要請があった．この要請は，Forte 経験者が全くいない関西側の責任者の立場に立てば至極当然であり，東西拠点に対する技術支援の公平性を確保する意味でも，できるかぎり要請に応えるべきと考えた．しかし，当時，3〜4 名で編成された技術支援チームでは対応が困難なため，

外部コンサルタントの導入や，当社スタッフからの増員などを検討したが，要員確保の問題等から技術支援チームの主力メンバ 1 名が，当分の間，週 3 日ペースで出張支援することで合意した．

実際の関西支援では，支援要請のほとんどは基盤チームからのものであった．基盤チームは, システム全体で共通的に使用する基盤ソフトウェアの開発を担当していた．基盤ソフトウェアの多くは，メインフレームと UNIX ワークステーションで動作していた旧システムの機能を踏襲するもので，構造的に特殊かつ複雑という特徴があった．基盤チームに対する具体的支援の例を以下に示す．

① システムの内部で非同期に発生する情報を，特定あるいは不特定多数のクライアントに同報通知するための「配信機能」のアーキテクチャ考案および設計例提示．

② システムの部分更改や障害発生などの際に，システムが提供するサービスの一部を閉塞するための「業務規制機能」のアーキテクチャ考案および開発．

③ 通信設備に対する制御コマンド送信や応答受信などを行なう「他システム接続機能」のアーキテクチャ考案および設計，開発．

④ クライアント PC 上で動作する各サブシステムのクライアントプログラムを統括，制御する「クライアント統合機能」のアーキテクチャ考案および実装サンプル提示．

⑤ クライアントプログラムとサーバプログラムのバージョンの整合性を監視し，不整合がある場合は，自動的にクライアントプログラムの更新を行なう機構の検討．

このうち，①〜④は最終的に実現したが，⑤は機能としてあまりにも複雑過ぎるため，実現は断念した．

上記作業の中で，アーキテクチャ考案とサンプル提示は関西支援担当 1 名で対応したが，設計例提示や予定外の開発受託については，技術支援チームで設計・開発スタッフを増員し対応した．

4.6 技術質問回答サービス

関西・基盤チームへの支援が一段落したころ，関東と関西を結ぶ Notes ネットワークが開通した．Notes ネットワークで，文書共有やメールによる情報交換が可能になったことで，関西と関東に対する技術支援は，原則としてすべて Notes ネットワークを通じて行なうことになった．

表 2 に，技術質問回答サービス提供にあたっての基本方針と，それらの方針で活動した結果，得られた（と思われる）効果を示す．

これらの方針の中で，特に重視したのが No. 3 の方針である．一般に，オープン製品では，製品のソースコードまで入手できることは希であり，通常は製品自体がブラックボックスになっている．そのような製品で発生する問題の根本原因を特定し除去することは現実的には不可能に近い．しかし，だからといって，製品に絡む問題をすべて製品メーカに頼るのでは，開発チームからの信頼も得られない．

なお，No. 4 の方針に従って提供した情報やツールの代表的なものとしては，以下のものがある．

表 2　技術質問回答サービスの方針と効果

No	方針（上段）と効果（下段）
1	すべての質問と回答は、質問者だけでなく、プロジェクトに参加している全ての企業に公開する。
	技術支援が全ての参加企業に対して公平に行なわれていることを目に見える形で示したことで、参加企業間の誤解や摩擦を未然に防止できた。
2	質問や相談の回答は、文章だけでなく、できるだけ実装例（サンプルプログラム）を添付する。
	問題の核心部分だけでなく、周辺のちょっとしたテクニックなども同時に伝えられるため、Forte に不慣れな開発者にとってはかなり参考になったものと推測される。また、回答文の曖昧さを補完する効果もあった。
3	トラブル報告については、次の優先順に従って、問題が解決するまで関与する。 1) トラブルの発生原因を突き止め、原因自体を取り除く。 2) 発生原因が不明であるか、原因の除去が困難な場合は、回避策を提示する。 3) 原因も回避策も不明の場合は、トラブルの発生確率を下げる予防手段を提示する。（運用等による回避策等）
	各社と問題を共有し、最後まで支援を提供することで、各社の技術支援チームに対する信頼を高める効果があった。
4	質問や相談に関連して、必要と考えられる情報やアドバイス、ツール等は、要請が無くても積極的に発信する。
	教育やトレーニングの不足を補う効果があった。また、情報提供にあたっては、プロジェクトの成功に不可欠と判断した情報は、たとえ他社に対してであっても惜しむことなく、持てる情報の全てを提供するようにしたことで、技術支援チームに対する各社の信頼を高める効果もあった。

① Forte による高性能システム開発のヒント

Forte で高性能システムを開発するための 29 項目のポイントの解説資料

② Forte における分散トランザクションの考察

分散トランザクションシステムで 2 PC（Two Phase Commitment）を採用すべきかどうかに関して，障害発生確率と対策コストの観点から考察した資料

③ 端末情報等の持ち回りの一手法

ログイン ID 等の情報を，一連の処理（タスク）が完結するまで，分散環境内のどこででも参照できるようにする手法の紹介資料

④ メッセージカタログのアクセス方法比較

エラーメッセージの格納と表示方法について，3 通りの方法を紹介し，その中の一つを推奨した資料.

⑤ Forte 社から様々なルートで入手した技術情報

膨大な技術情報の中から，マニュアル記述の無い有益な情報を適宜，選択し提供.

⑥ GhostBuster ツール―肥大化したプロセスサイズの 60～70% を削減するツール

⑦ StepCounter ツール―Forte ソースのメトリックス（行数等）測定ツール

⑧ SupplyViewer ツール―Forte ソース間の関係をグラフィカルに表示するツール

5. 遭遇した主な問題と対策

製造工程の期間は，計画の遅延もあって実質的に 2 ヶ月強であった．その短期間で 250 万ステップのシステムを Forte 初心者だけで構築できたのは，前章で述べた技術支援の効果もさることながら，Forte の高い完成度に拠るところが大きかった．この意味で，プロジェクトが Forte を採用したことは間違いではなかった．しかし，その一方で，約 150 件もの相談，質問，トラブル対応依頼が技術支援チームに寄せられた．これらの相談等への対応が，プロジェクトの中盤から終盤にかけての技術支援チームの作業の大半を占めた．

特にトラブル関連の相談は，軽微なものから致命傷になりかねない重大問題まで，多岐にわたったが，幸いにして，それらのすべてを解決，回避，あるいは緩和することができ，大事には至らなかった．これらのトラブルのすべてを報告するのは，紙面の関係で無理であるが，以下の節では，その中でも比較的，重大であった幾つかの問題について，その内容と実際に採った対策を報告する．

5.1 ソースコード消失問題

【問題】

正当性を確認済みの Forte ソースを開発リポジトリからエキスポートし，別の開発リポジトリにインポートする際，ソースコードが全角文字を含んでいると次のような現象が起きる場合がある．

① ソースコードの一部が消失する．
② 正常な行で意味不明のエラーが出る．エラーは空白や TAB を挿入することで消滅することがある．
③ メソッドの実在する行が実行されない．
④ プログラムをコンパイルして実行すると異常終了する．

【対策】

リポジトリのインポート機能の日本語処理の欠陥が原因と判明した．Forte 社コンサルタントを呼び，回避ツールを開発させた．同ツールをプロジェクトに提供することにより問題は解決した．

5.2 DDE 同期喪失問題

【問題】

DDE（Dynamic Data Exchange）[*3] よるプロセス間通信の同期がとれなくなり，結果としてクライアントがハングアップする現象が開発中から頻発した．

【対策】

当初，発生頻度が不安定で再現パターンがなかなか掴めず，調査が思うように進展しなかった．その後，技術支援チームが，一定の確率で現象を再現できるサンプルプログラムの開発に成功し，代理店に提出した．代理店でも現象が再現し，サンプルは Forte 社にも送られたが，Forte 社では再現せず，調査は暗礁に乗り上げた．一方，開発チームでは，発生パターンの追求と回避策の発見に努め，独自に発見した幾つかの回避策で発生頻度をかなり抑えることに成功したが，撲滅するには至らなかった．最終的には，Forte 社コンサルタントに対応を依頼した．コンサルタントは，技術支援チームが開発した再現率の低いサンプルを，繰り返し実行するよう改造することで

再現率を高め，Forte 社に送付した．Forte 社でも現象が再現し，Forte の DDE 機構の不具合であることが確認された．不具合の原因もほどなく解明され，後日，改修版を入手した．

5.3 開発リポジトリ破損問題

【問題】

開発や試験の最中に幾つかの不可解な症状が発生した．例として次のような現象がある．

① アプリケーションのコンパイル時に UNIX C++コンパイルで意味不明のエラーになる．

② 開発環境で，開発したアプリケーションを起動すると意味不明のエラーになる．

【対策】

ガベージの堆積による開発リポジトリのフラグメンテーションが原因と判明したため，開発リポジトリの再構築を定期的に行なうよう指示した．

5.4 コンパイル版作成時の諸問題

【問題】

結合試験をコンパイル版で行うため，各社がコンパイル作業に着手したところ，下記の問題がほぼ同時期に噴出した．

① 肥大したクライアントを分割し，一部を DLL（Dynamic Link Library）[*4] 化することを試みたが，様々なエラーが頻発しコンパイルが成功しない．

② Forte 環境変数で与える最大オブジェクトメモリ値が小さ過ぎ，コンパイルが途中で異常終了する．

③ UNIX のカーネルパラメタで設定されているデータセグメントの最大値が小さ過ぎ，コンパイルが途中で異常終了する．これを回避するため，カーネルパラメタを最大値に変更したところ，プロセス間通信ができなくなった．

④ 1 台のサーバ上で複数の Forte 環境や Oracle を実行したため，メモリ不足でコンパイルが異常終了する．

⑤ 300 近いカラムを持つ巨大表を検索するための，長大な SQL（Structured Query Language）文を持つプロセスのコンパイルで，UNIX C++コンパイラが異常終了する．

⑥ VC++コンパイラがヒープ領域不足で異常終了する．

これらの問題は，もたらした混乱の大きさの点で，プロジェクトにとって最大の危機であった．

【対策】

緊急避難として，コンパイル版の作成は断念し，試験はインタプリタモードでおこなう方針に変更した．方針変更後は，試験のための出荷作業と試験は共に順調に進んだ．また，試験と並行して，個々の問題に以下のような対策を講じた．

① DLL 化は行なわないことを決定

② Forte 環境変数で与える最大オブジェクトメモリを最適化

③ UNIX カーネルパラメタを最適化

④　関西の環境 G が独自に対応
⑤　UNIX C++コンパイラをバージョンアップ
⑥　VC++のデフォルトヒープサイズを変更し最適化

また，各種メモリ不足問題の根底にあるパーティションの肥大化傾向に対して，技術支援チームが 4.6 節⑥で述べた不要コード削減ツール GhostBuster を独自に考案・開発し，プロジェクトに提供した．同ツールの適用により，プロセスのソースコード量を 60〜70% 削減することに成功し，プログラムの肥大化問題は劇的に改善された．製品試験工程の中盤で，これらの対策を全て適用した上で，コンパイル版の作成を再度試みた．対策は奏効しコンパイルに成功した．

5.5 性 能 問 題

【問題】

製品試験の最後に行なった性能確認試験で，応答が異常に遅い処理があることが判明した．調査の結果，以下の問題点が指摘された．

①　クライアントの起動から初期画面表示までの時間が長い．
②　アプリケーションサーバの CPU 消費が異常に多い機能がある．
③　プロセス間のメソッド呼出のパラメタとして数千個のオブジェクトを渡すと，プロセス間の転送処理だけで約 20 秒かかる．
④　ログ出力を 1 万回おこなうプログラムを作成し，Forte の環境コンソール[*5]でログ出力を可（Enable）にした場合と，不可（Disable）にした場合で処理時間を計測したところ，前者は 25 秒，後者は 17 秒であった．後者は物理ログファイルへの書込みは行なわず，出力要求は空振りで終わっているが，それでも実際に書き込んだ場合（前者）の 68% の時間が消費されている．

【対策】

問題が表面化したのがサービス開始目前ということもあり，B 社が緊急対策会議を召集した．技術支援チームも参加し，個々の問題について原因推定と対策検討を行なった．

①　原因：クライアントの実行ファイルサイズが大きく，メモリ・ロードに時間がかかる．
　　対策：GhostBuster ツール（4.6 節⑥参照）を適用し，クライアントプログラムのサイズを削減する．
②　原因：アプリケーション設計とプログラミングの問題か．
　　対策：アプリケーションコードの見直しと無駄の徹底排除．
③　原因：ポインタで結合された 3 次元構造を持つオブジェクトを，別パーティションに転送する場合，送信側の Forte は 1 次元構造に変換して転送し，受信側ではそれを再組立てする．20 秒の大部分は，数千個のオブジェクトの変換・再組立て処理とそれらの転送の時間と考えられた．1 個当りでは約 5.3 ミリ秒になる．
　　対策：数千個ものオブジェクトを 1 回のメソッド呼出しで渡す正当な理由があるなら，送信側と受信側のプロセスを統合しプロセス間通信を無くす．

④ 原因：各プロセスに内在する Forte のログ出力オブジェクトに制御が渡り，ログ出力オブジェクト内でログ出力の可/不可を判断し，書込み不要として制御を戻すための時間が予想以上に大きい．ログ出力 1 回当りでは 1.7 ms になる．

対策：不要なログ出力コードを削除する．特に，呼ばれる回数が多くなりがちな汎用部品的メソッドは，試験で品質確認後，ログ出力コードを削除する．

会議では上記対策案を出したが，真の原因の追求と対策は，開発を担当した各社に委ねられた．各社で様々な改善が行なわれ，最終的には，それらが奏効して，サービス開始後は遅いという話は殆ど聞かれなくなった．なお，後日談として，あるチームでは SQL の発行方法を，静的 SQL[*6] から，より高速な動的 SQL[*7] に変更したことで応答時間が大幅に改善できたとの情報もあった．結局のところ，原因の大半は設計とプログラミングの考慮不足ではないかと推測された．

5.6 Forte Keep Alive 問題

【問題】

サービス開始後，データベースサーバの CPU ボード不良が数回，Oracle のソフトウェアダウンが 1 回発生した．このとき，システム全体は停止せず，原因を取り除いたあと，データベースサーバだけを再立ち上げした．数日後，特定の機能の処理が遅くなり始め，日数の経過とともに遅延する機能が増え，同時に遅延時間も拡大していった．最終的には，システムの大半の機能が無応答状態に陥り，実質上のシステムダウンに至った．

【対策】

調査の結果，ダウンは，いずれも Forte Keep Alive 機能[*8] の不具合が 2 次原因であった．当面の対策として，本番環境全体への Forte Keep Alive の一律適用をやめる指示を発出した．なお，Forte 社は，次期バージョンでこの問題を改修することを表明している．

6. 技術支援の課題に対する実績評価

前章で挙げたような様々な問題に遭遇しつつも，プロジェクトは当初計画通りシステムを完成し，本番開始に漕ぎ着いた．プロジェクトを終えてみて，4 章の冒頭で掲げた技術支援の五つの課題は，結果としてどうであったのかを評価してみる．

課題 1. Forte とオブジェクト指向方法論に馴染みの薄い大量の開発メンバを，如何に素早く立ち上げるか．

評価 1. Forte については，教育や技術移転，随時おこなった各種支援が奏効し，大規模システムを予定通り完成できたことで，課題は達成できたと考える．

課題 2. 企業文化の異なる複数の企業に対して，如何にして公平な技術支援を提供するか．

評価 2. 当初，関西側開発チームから，技術支援に対する不満が出ていたが，その後，関西側を重点的に支援したことによって，プロジェクトの後半ではかなり信頼を得ることができた．支援の公平性という点では，むしろ関東の当社の開

発チームに対する支援が手薄になってしまったが，幸いにして同チームからは，特に不満の声は上がらなかった．文字通りの「公平な支援」をプロジェクト全体に提供できたかと言えば，答えは NO になるが，不満が生じ易い遠隔地の他社の支援に重きを置いたことで，「公平な支援」の上位目標である「円満なプロジェクト運営とプロジェクトの成功」は達成できたと思う．この意味で，課題はクリアできたと考える．

課題 3. 関東，関西の物理的な距離の隔たりを如何にして克服するか．

評価 3. 3 名の技術支援要員の常駐を希望し，最後まで希望が叶えられなかった関西側が，結果として充分な技術支援が得られたと考えているかどうか評価のポイントになる．実際の支援では，距離を克服する手段として，次のような対策をとった．

① 初盤 1〜2 ヶ月間の週 3 日ペースでの出張支援
② 中盤以降の Notes ネットワークを通じた遠隔支援
③ 終盤でのスポット的出張支援
④ 重大問題発生時の電話会議[*9] での支援

これらの手段の中で，特に威力を発揮したのが②の Notes による遠隔支援であった．この手段が無かったら恐らく関西側の不満が爆発し，2〜3 名の長期常駐支援は避けられなかったと思われる．結果として距離を克服した技術支援が提供できたかどうかは，関西側に判定して戴くしかないが，技術支援チームとしては，複数の手段を通じて関西側を重点的に支援したことで，満点は無理としても，及第点は戴けるのではないかと考える．

課題 4. 技術支援チーム自身にも経験の無い，Forte による大規模分散システムの開発を如何にして指導し成功に導くか．

評価 4. 結果論としては，システムが納期通り完成し，依頼主からも高い評価を得ているので，課題は達成できたと考える．しかし，開発の全工程を通じて適切な指導ができたかとなると別問題である．技術支援チームとしては，できる限りの支援を提供したつもりであるが，質，量とも不足したことはいがめない．その理由としては，完成したシステムに見られる次のような兆候が挙げられる．

① 完成システムの搭載機能の割にソースコード量が多い．
→ コードが洗練されていない可能性がある．
② Forte の特性を理解していないと思われるコードが随所に見受けられる．
→ プロトタイプの技術移転で説明した内容を，意味を理解せずに単純に真似たと思われるコードがある．実際の開発では，ケースバイケースで柔軟に対応する必要があるが，そこまできめ細かい指導ができていない．
③ システム全体のプロセス数が多過ぎる．
→ プロセス分割の考え方や，プロセス数増大のデメリットが，充分，指導できていない．

支援の質・量が不足した最大の原因は，システムの規模とプロジェクトメンバの総数に比べて，技術支援メンバの人数があまりにも少な過ぎたことがある．今回の支援要員は，単純計算では約 100 名に対して 1 名（1%）程度しかアサインできなかったが，初めて Forte を使うプロジェクトという条件を考慮すると，最低でも 10 名に 1 名（10%）程度は必要であったかもしれない．

課題 5. メンバが僅か数名の技術支援チームで，大規模プロジェクトに対する技術支援を如何に効率良くおこなうか．

評価 5. 充分にクリアした．プロジェクト全体からみれば，支援の質・量は不足していたかもしれないが，前述の評価 3 で述べた多彩な手段を駆使し，技術支援チームの実際のメンバ数以上の支援は提供できたと考える．

7. 今後の課題

今回の支援活動を終えてみて，当初，掲げた五つの課題もどうにかクリアでき，無事，本番を迎えられたことには満足感を覚える．しかし，途中で起こった様々な出来事を改めて思い起こすと反省すべき点も多い．

反省点の中でもとりわけ重要と思われることの一つに，フレームワークの不在がある．今回のプロジェクトでは，すべての機能をゼロから作り上げたが，こうしたやり方は，開発のスピード，生産性，品質，リスク回避，といった面で問題が多い．なるべく作らない，なるべく考えない（悩まない）で，要求されたシステムを素早く，安く開発するためのフレームワークの必要性を痛感した．

今回の技術支援を通じて，多数の部品やサンプル，ツール，技術文書などを作成，提供したが，それらをベースとして，更にアプリケーションの一部を自動生成するツールや，再利用性の高い部品などを新たに追加し，「高生産性フレームワーク」として体系化することが今後の課題である．

8. まとめ

Forte という初めて経験するツールを採用し，大規模開発に挑戦したプロジェクトに技術支援担当として参加するにあたって，課題をどう捉え，実際にどのような支援を提供し，どのような問題にどう対応したかを述べた．また，プロジェクトの結果を踏まえて，最初に掲げた課題それぞれについて，実績はどうであったかの評価を述べた．

技術支援という役割は，プロジェクトのリスク緩衝地帯であることが多い．リスクの質や量の予測が困難な場合は，緩衝機構の良し悪しがプロジェクトの成否に決定的な影響力を持つことすらある．しかし，技術支援の業務は前もって予測することが困難であり，プロジェクトマネジメントの中で，技術支援をどのように計画に組入れ運用するかの定まった方法論は存在しない．技術支援の良し悪しは，担当者の知識や経験だけでなく，仕事に対する考え方や取組姿勢といった極めて人間的な要素によっても大きく左右される．個々の人間に依存する要素は技術論としては扱い難いテーマであるが，本稿のような多くの事例報告の中から，共通項が発見され体系化されて，技

術支援という，重要ではあるが曖昧な作業が，少しずつでも理論的に裏付けられることを願っている．

最後に，プロジェクトに参加し苦労を共にした全てのメンバに，紙面を借りて謝意を表したい．また，本稿が，今後，大規模プロジェクトでプロジェクトマネジャや技術支援を担当する方々に少しでも参考になれば幸いである．

＊1　有限の開発用資源で，より大きな規模の本番環境と同等の環境を擬似的に実現する Forte 固有の機能．
＊2　アプリケーションの内部構造をプレゼンテーション層，ファンクション層，データ層の 3 層に分割する考え方．
＊3　Windows 95 が提供する，プログラム間通信手段のひとつ．
＊4　Windows 95 上で動作する複数のアプリケーションで同一の機能を共有するための手段．DLL として分離された機能は，それを利用するアプリケーションとは別個のモジュールとして構築，配布ができる．
＊5　Forte で構築された分散環境全体を管理・制御するための，Forte 固有のグラフィカル・システム管理ツール．
＊6　Forte 言語仕様に組み込まれた SQL 命令で，データベースに対する要求発行の都度，データベース側で SQL 文の解析が行なわれるため，使用法が簡単な反面，処理が遅いという欠点がある．
＊7　SQL 文を文字リテラルとして 1 度だけデータベースに渡してデータベースに解析を依頼し，以後，SQL の実行要求は解析済 SQL の実行を要求するやり方．プログラミングが複雑になる反面，2 回目以降の SQL 要求で，解析処理が不要になる分，処理が高速になるという特徴がある．
＊8　ネットワーク障害を検出する為の Forte 固有の機能．TCP KEEP ALIVE 機能とほぼ同等の機能．TCP KEEP ALIVE は通常 2 時間程度に設定されているが，通常，Forte Keep Alive は TCP KEEP ALIVE より短時間で障害を検出するために使用する．
＊9　遠隔地の会議室を電話で結び，出席者全員が自由に会話できるシステム．

参考文献

［1］　URL：http : //www.spr.com/library/0 langtbl.htm.
［2］　トレーニングマニュアル「Forte コンセプト V 2.0.1」,（株）シリウス 1996.9.4.
［3］　E. Gmma 他著“オブジェクト指向における再利用のためのデザインパタン”p. 197 ソフトバンク.
［4］　URL：http : //www.forte.com/support/white_papers.html
“Forte Design Patterns : Database Entity Manager”

執筆者紹介

菅　原　　伸（Shin Sugawara）
1971 年日本ユニシス（株）入社．汎用機通信制御システム開発，Videotex システム開発，表計算ソフト移植，オフコン客先システム開発，CAI 教室管理システム開発，オフコン用 NFS 開発，IRDS リポジトリエンジン開発，フリーキーワード全文検索エンジン開発などを経て，1995 年より Forte 客先適用に従事．現在，社公ソリューションサービス部テレコムシステム開発室所属．