HULFT for Mainframeを
継続的に価値向上するために

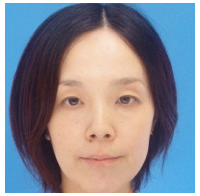
株式会社セゾンテクノロジー
HULFT開発部
清水麻梨子

自己紹介

株式会社セゾンテクノロジー
HULFT開発部Dev3課 課長　Mainframeチーム長

清水 麻梨子

- Mariko Shimizu -



HULFT
HULFT SQUARE
DataSpider
Servista

経歴

2004/04：セゾン情報システムズ入社（現在、セゾンテクノロジー）
2004/07：HULFT Mainframeチーム配属

HULFT6（Ver.603～）	※最初はUTL小規模修正、SMSクラス対応
HULFT7（初期～）	※RDW付集信、データ検証など
HULFT8（初期～）	※配信、集信、要求受付など全般

HULFT10 for zOS（2024/12リリース）
新機能DEFLATE圧縮の設計・実装メイン担当

メイン担当OS

- **z/OS（IBM社）、MSP／XSP（富士通社）**
- ACOS（NEC社）、VOS3（日立社）

開発言語

アセンブラ（HLASM）、COBOL、他

Mainframeチーム紹介

役割（一部）

- HULFT Mainframe製品の開発、リリース
- 最新z/OS対応検証（z/OS 3.2検証準備中）
- 問合せの調査、回答

体制

- 現在は10名程度（開発規模により増）
- 平均年齢：35～36才（20,30,40,50代すべての年齢層）
- 新入社員を数年に一度は必ず配属

今日お話しすること(3つ)

□HULFTで高速／高頻度／大容量転送を安全に実現

① zEDCを利用したDEFLATE圧縮

② ICSFを利用したAES暗号　※研究段階であり、本日のみ一部ご紹介

□HULFT Mainframe開発を30年以上継承する施策

③ Gitサーバを製品マスタにしたプロセス改善

今日お話しすること(3つ)

SAISON TECHNOLOGY

□HULFTで高速／高頻度／大容量転送を安全に実現

① zEDCを利用したDEFLATE圧縮

② ICSFを利用したAES暗号　※研究段階であり、本日のみ一部ご紹介

□HULFT Mainframe開発を30年以上継承する施策

③ Gitサーバを製品マスタにしたプロセス改善

HULFT10 for zOS-Enterprise(新グレード)

zEDC DEFLATE圧縮を追加

転送時間 転送速度が大幅向上！

圧縮なしと比較して**最大5倍短縮**

HULFT10 ⇄ 圧縮なし

↓ 5倍の転送効率

HULFT10 ⇄ DEFLATE圧縮

CPU使用率 コストを最適化！

HULFT8と比較して**最大70%削減**

HULFT8 ⇄ 圧縮なし

↓ CPU使用率70%削減

HULFT10 ⇄ DEFLATE圧縮

zEDC DEFLATE圧縮の効果的なユースケース

転送速度を上げて、転送時間短縮！（CPU使用率も減、コストも最適化）

ロ こんな要望に対応しました💡

✓「HULFTの転送速度を向上したい！」

✓「OS全体のCPU負荷を下げたい！」

✓「CPUランニングコスト最適化したい！」

ロ こんな運用ありませんか？💡

✓z/OS上の大容量データ利活用のため、データを高速に送りたい

✓日次夜間バッチがギリギリで後続処理に影響がある

■zEDCとは… zlib for zEnterprise Data Compression

- zCPU筐体のCPUを利用して、圧縮率の高いDEFLATE圧縮を実現できるz/OS機能
- z/OS上のCPU利用として扱われない

DEFLATE圧縮可否パターン

SAISON TECHNOLOGY

□相手先ホスト（配信先、配信元）がHULFT8のままでも、
zOS版をHULFT10にすることで、DEFLATE圧縮を利用可能

		相手ホスト（配信先、配信元）			
		zOS		Windows/Linux/UNIX/ IBMi/NSKなど	
自ホスト		HULFT10	HULFT8	HULFT10	HULFT8
zOS	HULFT10	可能	不可能	可能	可能
	HULFT8	不可能	不可能	不可能	不可能

DEFLATE圧縮でデータを小さくする意味

DEFLATE圧縮を使って

データが小さくなると、どんな効果があるか？

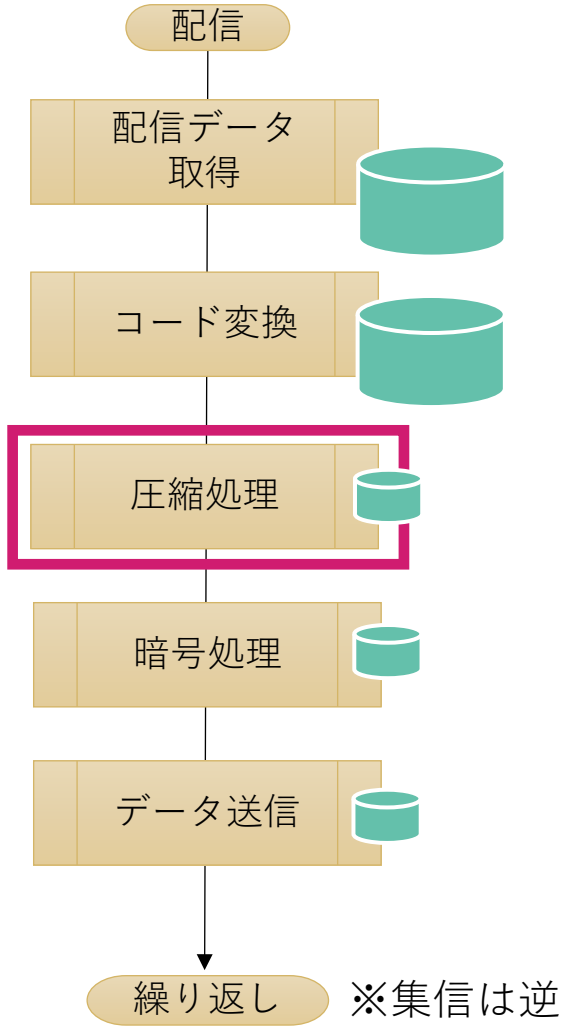
- **圧縮の後の暗号処理も速くなる**
- **ネットワークに流れるデータも小さくなる**

↓

- **集信側の暗号処理も速くなる**

↓

- **転送時間全体が小さくなる**



配信・集信双方で処理全体の速度向上が見込めます！

HULFT圧縮とDEFLATE圧縮

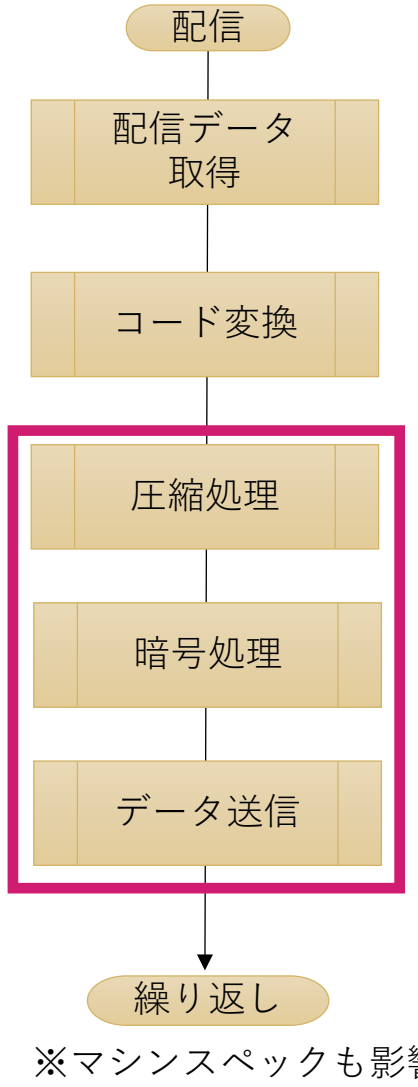
■ HULFT（横圧縮／縦横圧縮）

HULFT固有の圧縮方式

■横圧縮／縦横圧縮
- 連続した文字は小さくなりやすい
  （例：SPACE 0x40連続が多いジョブログデータなど）
- ただし、データにより圧縮エラーになる場合がある
  （配信エラーになる）

■ DEFLATE圧縮

zEDCのzlib関数による圧縮方式（OSS）



※マシンスペックも影響

DEFLATE圧縮（OSS）を利用することで、より圧縮効果が見込めます！

ただし、どちらも圧縮率は配信データ内容に依存します！

↓

データ内容だけでなく暗号、ネットワークなど、お客様環境より効果が異なります！

評価版について（無償）

評価版（HULFT10 for zOS、HULFT8 for zOS/MSP/XSP）を弊社HP「myHULFT」で公開しています
お客様環境でのデータ、ネットワークで効果をお試しいただけると幸いです

1 myHULFTにログイン

「myHULFT」にログインしてください。

2 製品のダウンロード

利用する製品のダウンロードと
プロダクトキー発行をしてください。
その後、製品のインストールが可能
です。

3 ライセンス情報の確認

ダウンロードした製品の
ライセンス情報とサポート情報など
が確認できます。

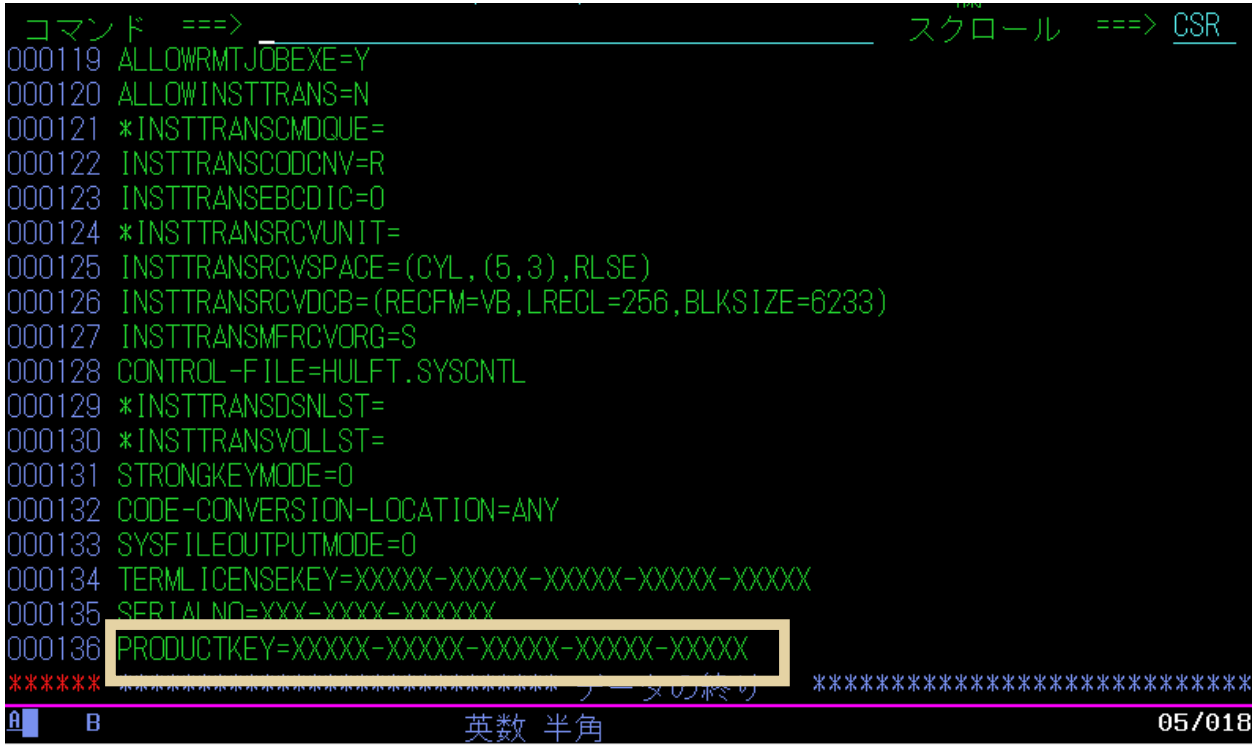
■対応OS
HULFT10 for zOS　：z/OS V2R5、3.1、3.2（2026/01検証準備中）

※動作環境はHULFT.comを確認

【補足】評価版→正式版への切替え

システム動作環境設定（PARMLIB内HULPRM）の、

プロダクトキー（PRODUCTKEY）変更のみで、評価版→正式版へ切り替えられます（再起動は必要）



ぜひ、ご検討のほど、よろしくお願いいたします

■製品グレード変更について
HULFT10 for zOS-Standard（DEFLATE圧縮なし）→ Enterprise（DEFLATE圧縮あり）の変更も、
プロダクトキー（PRODUCTKEY）変更のみで切り替えられます

今日お話しすること(3つ)

SAISON TECHNOLOGY

□HULFTで高速／高頻度／大容量転送を安全に実現

① zEDCを利用したDEFLATE圧縮

② ICSFを利用したAES暗号　※研究段階であり、本日のみ一部ご紹介

□HULFT Mainframe開発を30年以上継承する施策

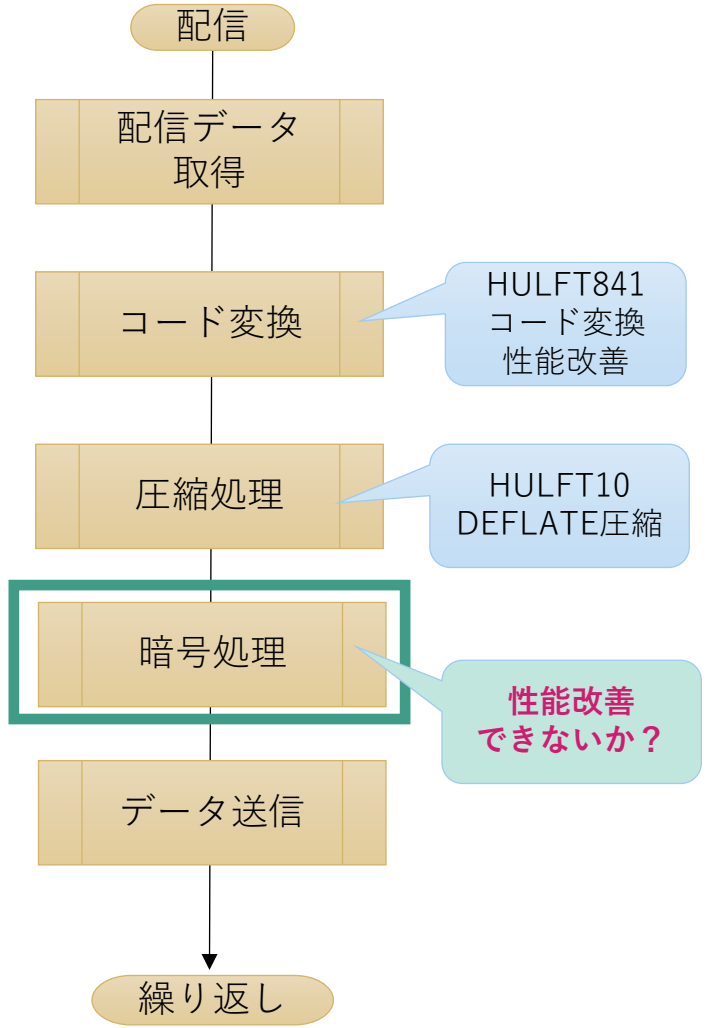
③ Gitサーバを製品マスタにしたプロセス改善

AES暗号の改善

AES暗号オプション利用時の転送データの暗号処理で、

転送時間、CPUの削減を見込める可能性があり、

研究を行っています



■ご留意事項
現在、製品への対応は未決定です
今後の調査より、製品への対応が見送られる可能性があります

本スライドは、

イベント当日限定にて

ご紹介した内容となります

（配布資料では割愛させていただきます）

今日お話しすること(3つ)

SAISON TECHNOLOGY

□HULFTで高速／高頻度／大容量転送を安全に実現

① zEDCを利用したDEFLATE圧縮

② ICSFを利用したAES暗号　※研究段階であり、本日のみ一部ご紹介

□HULFT Mainframe開発を30年以上継承する施策

③ Gitサーバを製品マスタにしたプロセス改善

HULFT Mainframeリリースの歴史

1993年にHULFT Mainframeをリリースし、32年以上開発を継続

■ 年表

年度		対応OS
1993	通信ミドルウェア「HULFT」提供開始	MSP版、OS390版、XSP版をリリース
1995	HULFT3 販売開始	VOS3版をリリース
1996	HULFT4 販売開始	
1999	HULFT5 販売開始	ACOS版をリリース
2003	HULFT6 販売開始	

■ z/OS

HULFT7 for zOS（2008/10リリース）

HULFT8 for zOS（2015/12リリース）

HULFT10 for zOS-Enterprise/Standard（2024/12リリース）

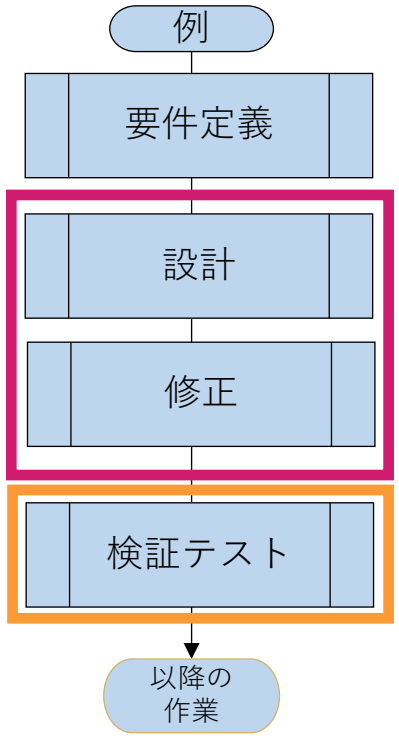
HULFT Mainframe開発の重要な観点

検証テストより前に正しいロジックにする（バグを発生させない）ことが重要です

HULFT Mainframe開発には、**重要な観点が多数存在**します

OS全体、HULFTシステム全体を考慮し、確認するには技術が必要です

【例】機能追加する場合
- **コード変換**結果へ影響ないか？
- **ジョブ実行**機能へ影響ないか？
- **集配信履歴**内容へ影響ないか？
- その他
  - 転送性能が劣化していないか？
  - コンソールメッセージ、完了コード設計は妥当か？
  - メモリ管理は適切か？
  - OS資源排他シーケンスは適切か？
  - プログラム／サブルーチン設計は適切か？
  - 他にも多数



※技術の習得には時間がかかります

旧来の開発手法と問題点

旧来はMainframe上で開発を行っていました

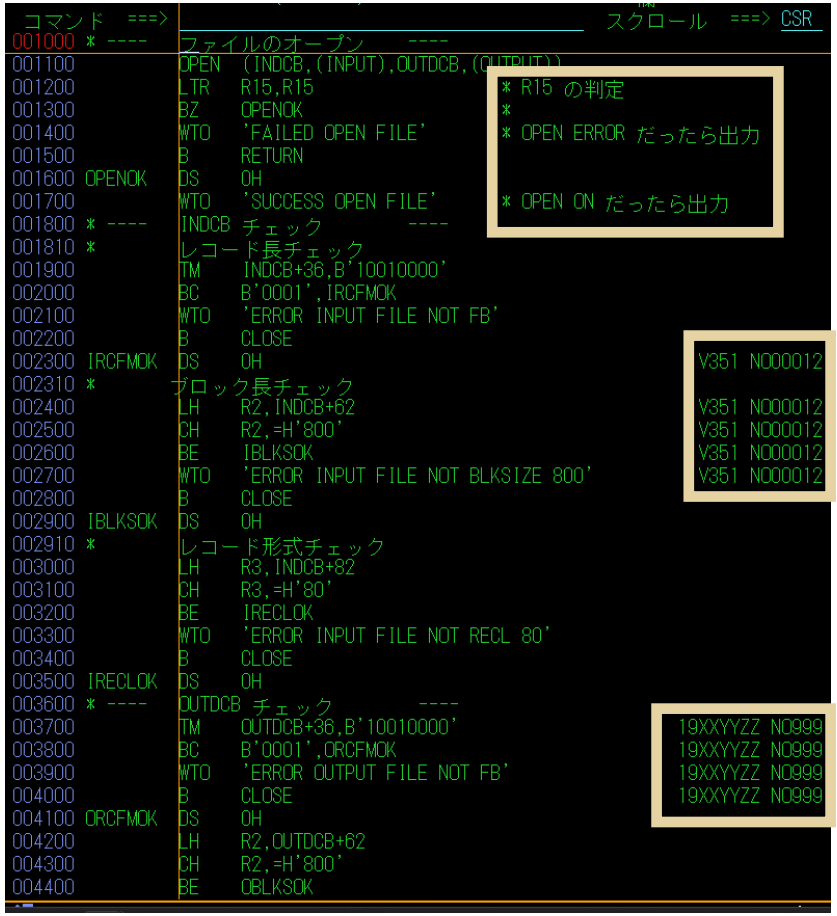
慣れていないと重要な観点を確認しにくい問題がありました

※Mainframeユーザのみなさまはイメージしやすい問題点だと思います

＜問題点の一例＞

- ソース全体が見にくい（解析しにくい）
- 複数の人間が直接1ソースを修正できない（ミス誘発）
- ロジックコメントが書きにくい（72バイトしかない）
- 他多数
  - 検索しにくい（3.12画面）
  - 比較しにくい（3.14画面）
  - どこを直したかわかりにくい（差分）

※z/OS上でのアセンブラソース例



改善後の開発手法（PC上での開発）

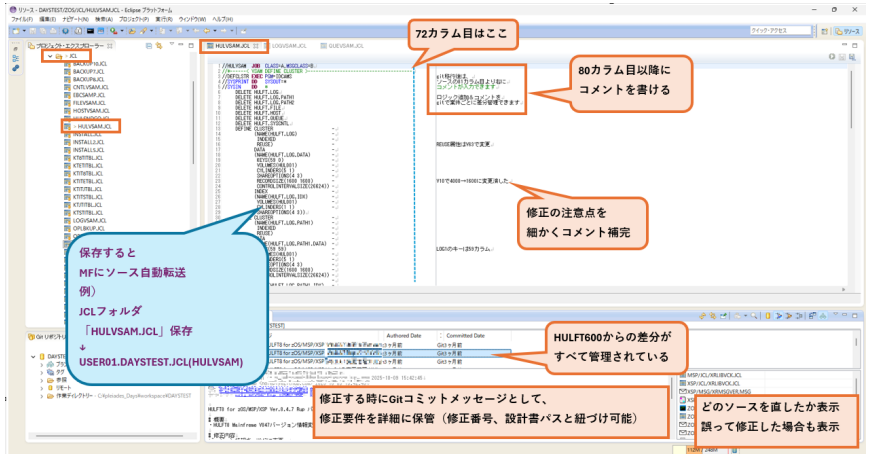
Gitサーバに製品マスタがあり、PC上でソース修正、差分をGit管理に変更しました（10数年前より）

開発品質（生産性）を大幅向上することできました

＜**問題点の解消**＞

- ソース**全体が見える**
- 複数の人間が直接1ソースを**同時修正可**
- ロジックコメントを**十分に書ける**
- 他多数
  - 検索→**大幅に改善**
  - 比較→**大幅に改善**
  - どこを直したかわかりにくい
    →**Gitで差分管理可、レビュー品質が大幅向上**

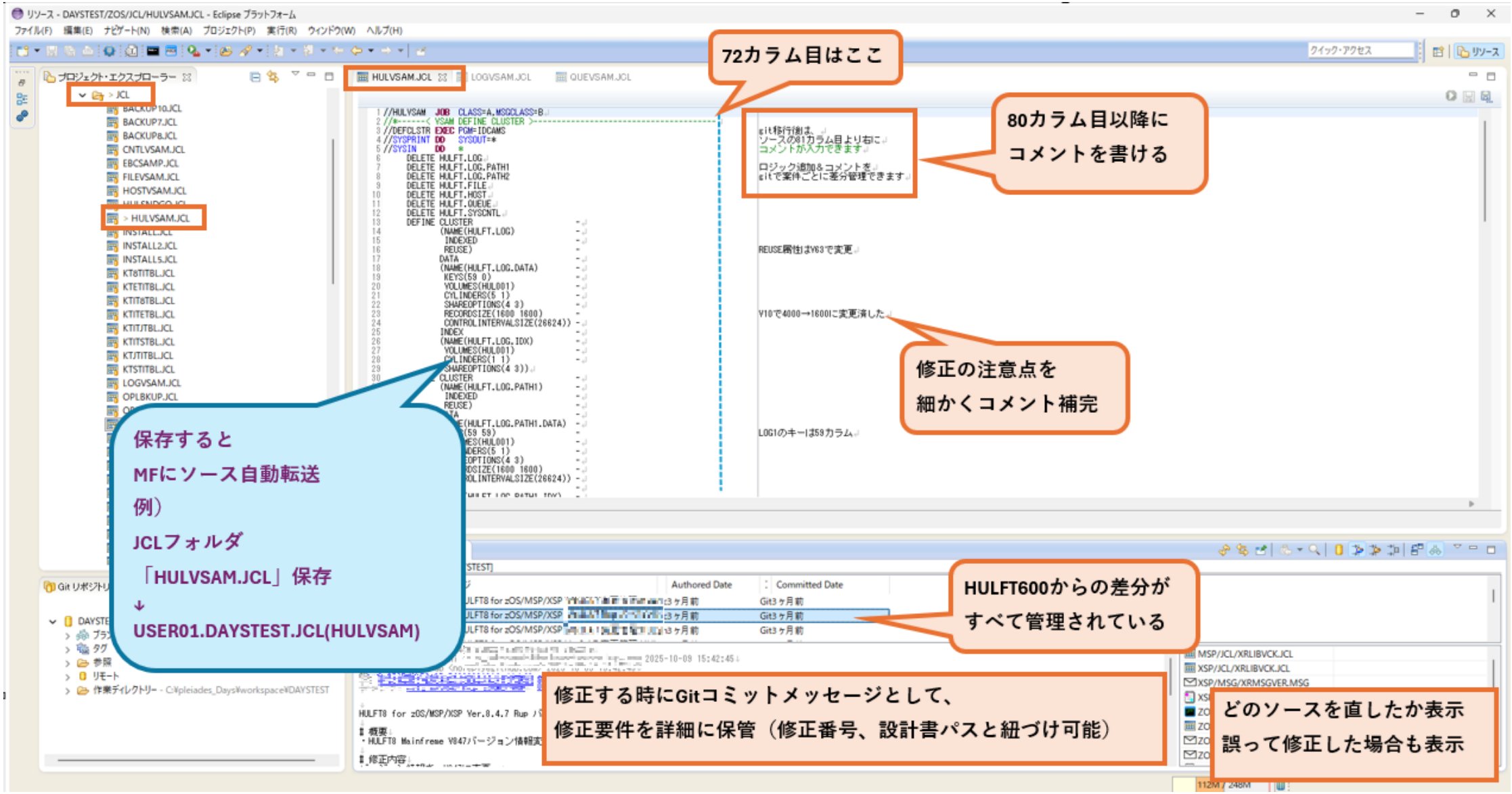
※次シートで拡大サンプルあり



環境向上→技術も向上

- **ソース解析力**
- **可読性の高いソースの生成**
- **レビュー品質の向上**
- **ヒューマンエラー防止**

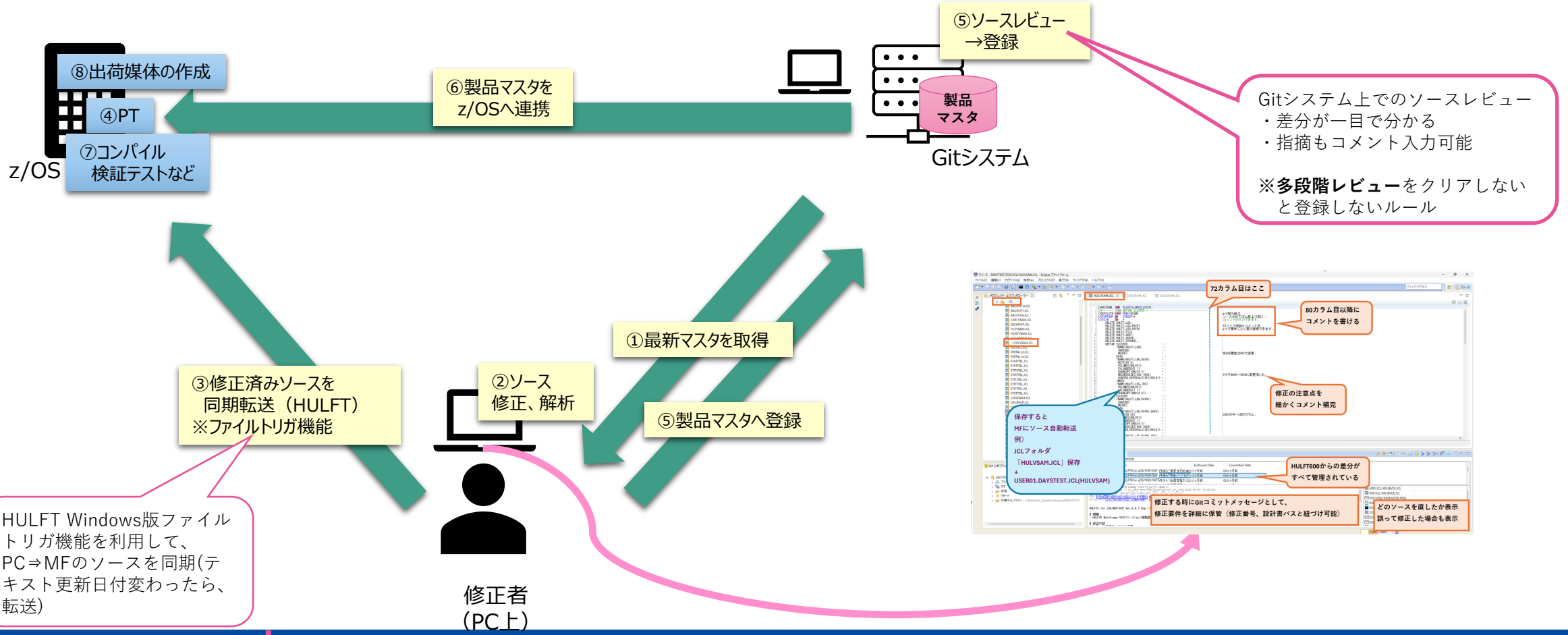
【サンプル】修正画面イメージ



【付録】MainframeとGIT構成図

修正担当者はPC上でGitに対し、修正を行います（レビューもGit上）

PC⇔Mainframeソースは、HULFT利用ツールで同期をとります（ファイルトリガ機能）



技術育成と継承（環境面以外の取り組み）

他にもさまざまな取り組みを行っています

前提として、業界全体の「**Mainframeかつアセンブラ技術者**」は少ないです

HULFT Mainframeは開発を続けてきました

□ 過去ナレッジ活用

- 古いマニュアルPDFを**ツールで一括検索**
- 設計書、構造図の拡充の**継続**

□ 若手メンバ技術スキル補完

- 重要な観点は「レビューシート」で確認し、**スキルを補完**

□ アセンブラ言語カリキュラムの充実

- **HULFT処理ベース**
  - データセット作成、入出力（DYNALLOC、OPEN/CLOSE / GET / PUT）
  - SYSUDUMP解読手法
  - アセンブラ命令マクロ構造化（アセンブラだけどIF／LOOP／SWITCH文）

最後に

□継承に最も大切なこと

□ HULFT Mainframeの重要性の継承

お客様運用で接続先がオンプレ、クラウド（AWSなど）、コンテナなど多様化、
レガシーデータ利活用（モダナイゼーション）など、
従来の転送処理だけでなく、様々な場面で使われ方が変わり進む中、
お客様環境でHULFT Mainframeが担うデータ転送の重要性を伝え続けていくこと
（高速／高頻度／大容量転送を、安全にご利用し続けていただく必要がある）

今後も新技術、性能向上など、研究を続けていく予定です

ご要望など、ぜひ伺えますと幸いです

本日はありがとうございました

＜免責条項＞

本資料の内容は、資料作成時点の当社の判断に基づいて作成されているものであり、今後予告なしに変更されることがあります。よって本資料使用の結果生じたいかなる損害についても、当社は一切責任を負いません。また、本資料の無断での複製、転送等を行わないようお願いいたします。なお、本資料に記載されている会社名、製品名は各社の商標または登録商標です。