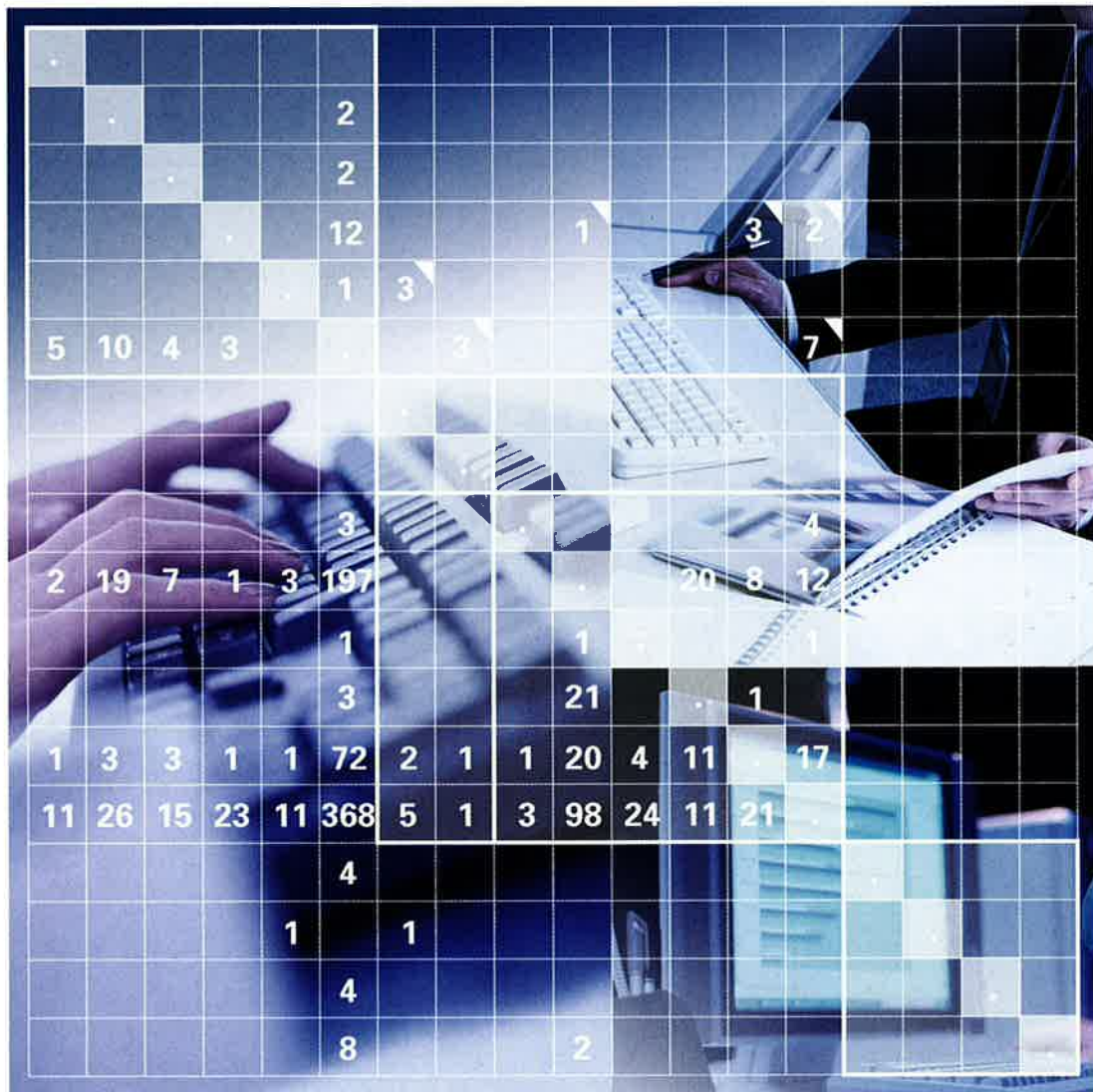


アーキテクチャ分析ツール

LATTIX

Ver.6.5



アーキテクチャの分析・改善に必携のDSM分析ツール

- 既存資産の再利用性と保守性の向上
- 同時並行開発の効率性アップ
- 影響分析の工数を大幅削減
- ソフトウェアの設計品質をモデルから分析・改善

複雑になったソフトウェア資産や設計図を Lattixがすっきり「見える化」します。

これまでの手法では、大規模で複雑なシステムやプログラムの構造を把握することは、非常に困難でした。DSMを用いたアーキテクチャ分析ツールLattixを使えば、C/C++で開発された組み込みソフトウェアや、Java、.NET、Oracle、SQLなどで構築された大規模システム、さらにはUML/SysMLなどで記述されたモデルなどを直感的でわかりやすい表形式に置き換えることができます。Lattixはこれらの全体像を可視化し、その構造や依存関係を正確に把握するために最適のツールです。ソフトウェア設計品質の改善、ソフトウェアプロダクトラインの推進、トレーサビリティの確保などの難題に取り組まれているお客様を、Lattixは強力にサポートします。

*DSM: Dependency Structure Matrix

LATTIX

DSMを活用してアーキテクチャを分析

開発プロセスにおける様々な問題

システムが巨大化・複雑化し、
全体の構造が把握できない

当初の設計と実装が
乖離してしまっている



構造が複雑で、修正の
もぐら叩きになっている

機能分割できていないため、
効果的な再利用ができていない

変更を加えた場合に
影響が及ぶ範囲が分からない

DSMのアーキテクチャ分析への応用

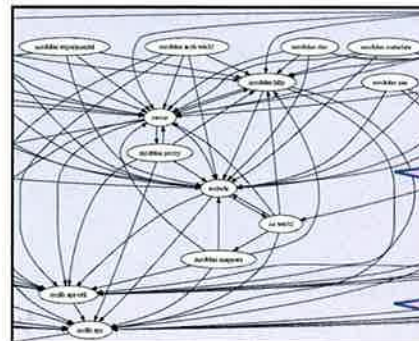
Lattixは、DSMアプローチをベースにしています。DSMとは、Dependency Structure Matrixの略であり、繰り返しやフィードバックが多い開発プロセスをわかりやすく表現するために開発された、プロセス改善のための分析技法です。DSMは、1968年にDonald Steward氏によって開発され、以後、マサチューセッツ工科大学、ハーバード大学、イリノイ大学などの数多くの大学で研究が続けられ、ここ10年の間、Boeing、Lockheed Martin、Intelなどの企業において、多様な分野で幅広く用いられてきました。DSMを利用することで、これまでは分析が困難だったシステムの依存関係を、より直感的に可視化・分析することが可能になります。

ModuleBはModuleAとCから**依存されている**

	Module A	Module B	Module C	Module D
Module A	1			X
Module B	2	X	X	
Module C	3	X	X	
Module D	4		X	

ModuleCはModuleBとDに**依存している**

UMLなどでのリバースモデリング分析



ファイル間の関連が
描画された巨大な
Box Diagram

アーキテクチャ(設計構造)
を把握し、検証することは
現実的には不可能

DSMによる分析

	support	os.win32	modules	server	include	src.lib
\$root	1	2	3	4	5	6
+ support	1					
+ os.win32	2		2		2	
+ modules	3	2		12		
+ server	4	1	1	332		
+ include	5	3	4	528	158	
+ src.lib	6	52	4	510	142	48

階層マトリクスによる表現で、
巨大なソフトウェアで
あっても全体を俯瞰可能

階層化の妥当性、設計
との乖離を把握可能



Lattixを使ったソフトウェアアーキテクチャの分析例

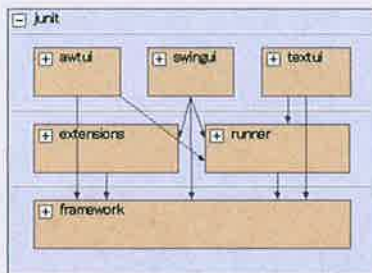
分析例1：JUnit

DSM図

\$root	textui	swingui	awtui	runner	extensions	framework
1	1					
2		1				
3			1			
4	5	12	3	1		
5					1	
6	8	19	4	6	12	1

パッケージ階層をLattixで分析した結果。パッケージ間では循環の依存関係がなく、良好な状態と評価できる。

アーキテクチャダイアグラム図



Lattixでアーキテクチャダイアグラム図に変換。UI層、ロジック層、フレームワーク層と3層構造にパッケージが階層化されている様子が確認できる。

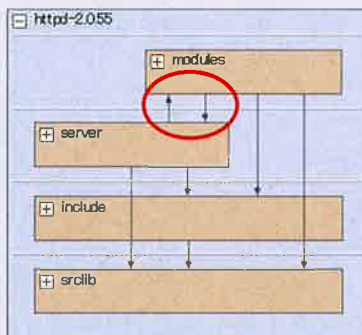
分析例2：Apache httpd

DSM図

\$root	modules	server	include	src/lib
1	1	12		
2	332			
3	528	158		
4	510	142	47	

modulesとserverの間に12箇所の循環依存を発見。

アーキテクチャダイアグラム図



下位階層から上位階層への依存が存在しており、当初設計と乖離している。

〈DSM図をさらに展開〉

\$root	aaa	archwin32	cache	dev	echo	experimental	filters	generators	http	loggers	mappers	metadata	proxy	server
1	1													
2		1												
3			1											
4				1										
5					1									
6						1								
7							1							
8								1						
9									1					
10										1				
11											1			
12												1		
13													1	
14														1
15	26	6	6	18	2	43	17	40	2	3	43	2	24	26

serverから、modules/httpとmodules/proxy への循環依存

DSM図を展開すれば、ソースファイルやヘッダーファイル、関数、変数といった、更に詳細な粒度での可視化が可能。

ソフトウェアの再利用性、保守性の向上と開発工数の削減に役立つさまざまな機能

■ 設計規則の活用

Lattixでは、アーキテクチャの品質を長期的に維持するために、設計規則を設定しチェックするための機能を備えています。DSMマトリクス上に設計規則を設定しておくことで、一目でアーキテクチャ上の問題点が確認することができます。また、定期的に設計規則違反をチェックすれば、アーキテクチャの乱れを抑制することが可能です。

\$root	Subsystem1	Subsystem2	Subsystem3	Subsystem4
1	1			
2		3		
3			1	
4	6	4		1

緑色の三角マーク：依存関係が正しい
黄色の三角マーク：依存関係があてはまらない
赤い三角マーク：設計規則に違反する

■ 影響範囲の分析

Lattixは、変更する要素(関数、変数、クラスなど)にマークをつけるだけで、指定した要素が変更された際に影響を受ける他の要素の一覧をリストアップします。機能追加やバグ修正による影響範囲を容易に把握できるため、変更作業に伴うリスクをコントロールすることが可能になります。



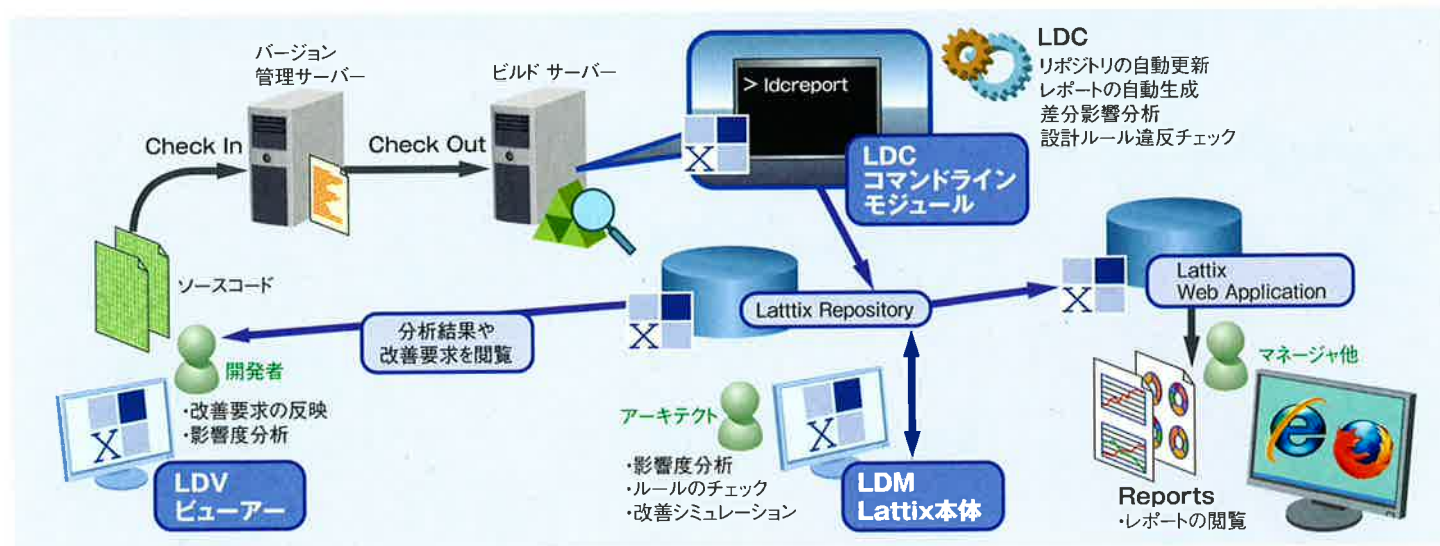
■ メトリクスの算出と分析

ソースコードの品質を定量的に評価するソースコードメトリクスは世に広く知られ利用されています。しかし、ソフトウェアアーキテクチャを定量的に評価するのは非常に難しい問題でした。Lattixでは、システムの複雑度や結合度、循環度など、約30種類のメトリクスを使って、ソフトウェアアーキテクチャを定量的に評価することが可能です。

測定するメトリクス

システム規模	連結の度合い	循環度
・要素数	・正規化累積依存度	・循環度
・複雑度	・連結度	・階層循環度
・依存関係数	・連結濃度	・システム間循環度
・依存度平均	・連結強度	
変更に対する感度	結合の度合い	
・システム安定性	・結合度	
・影響度平均	・結合濃度	
	・結合強度	

■ システム運用イメージ



■ Lattixの稼動環境

● Lattixクライアント稼動環境

OS	Windows 2000 / XP / 2003 / Vista / 2008 / 7 (32bit / 64bit) Linux (x86 / x86-64) ※ 64bit環境は、分析対象により対応していない場合があります。 ※ VMWareなど、仮想OSには対応していません。
メモリ	512MB以上 (1GB以上を推奨)
Java	JRE (Java Runtime Environment) 6.0 以上
Javaヒープサイズ	1GB以上を推奨

● Lattixライセンスサーバーの稼動環境

OS	Windows 2000 / XP / 2003 / Vista / 2008 / 7 (32bit / 64bit) Linux (x86 / x86-64) ※ VMWareなど、仮想OSには対応していません。
----	---

* Understandデータを分析するには、Lattixがインストールされているマシンに、Understandおよび有効なUnderstandのライセンスが必要です。
 * Klocworkデータを分析するには、Klocworkデータベースサーバーへのネットワーク接続が必要です。
 * Rhapsodyデータを分析するには、Lattixがインストールされているマシンに、Rhapsodyおよび有効なRhapsodyのライセンスが必要です。
 * Oracle, SQL Server, Sybaseデータベースを分析するには、各データベースサーバーへのネットワーク接続が必要です。

● 分析対象

分析対象	分析対象ファイル
Java	Java jar/war/ear, class, zip Hibernate hbm, hbm.xml, cfg.xml Spring xml
C/C++	BSC Visual Studio C/C++ BSC (*.bsc) Doxygen doxygen xml (1.5.1以降) Understand Understand 1.4 for C++ Database (*.udc) Understand 2.0/2.6 Database (*.udb) Klocwork Klocwork Insight 8.2/9.0 Database
.NET	.NET dll, .NET exe (Managed)
UML/SysML	Rhapsody Rational Rhapsody Project (*.rpy) (7.2以降) XMI XMI 2.1 (XMI 1.2 は部分的にサポート)
ActionScript	swf, sfc, link-report.xml
Ada	Understand 1.4 for Ada Database (*.uda) Understand 2.0/2.6 Database (*.udb)
Delphi Pascal	ModelMaker Tools Pascal Dependency Analyzer (*.ldp)
Fortran	Understand 2.0/2.6 Database (*.udb)
Oracle	Oracle 9i/10g/11g Database
SQL	SQL Server 2000/2005/2008 Database Sybase ASE (Adaptive Server Enterprise) 12.5.x/15.x SQL File, DDL File
LDI (Lattix Data Import)	Idi, Idi.xml

【開発元】

LATTIX

Lattixの情報は <http://www.techmatrix.co.jp/quality/lattix/>

- 本カタログに記載されている内容は予告無く変更される場合がございます。
- 掲載されているあらゆる製品名は、各社の商標あるいは登録商標です。

【総販売代理店】

TechMatrix

テクマトリックス株式会社

システムエンジニアリング事業部 ソフトウェアエンジニアリング営業部
 本社: 〒108-8588 東京都港区高輪 4-10-8 京急第7ビル
 TEL.03(5792)8606 FAX.03(5792)8706
 大阪営業所: 〒541-0054 大阪市中央区南本町 2-6-12
 サンマリオンNBFタワービル
 TEL.06(6243)3801 FAX.06(6243)3803
 【URL】<http://www.techmatrix.co.jp>
 【E-mail】lattix-info@techmatrix.co.jp

R100

合成顔料100%完全無害
使用しています。



このカタログの印刷には、環境に配慮した
植物性インキを使用しています。

Lattix ユーザー事例 RICOH

組み込みソフトウェアの依存関係を可視化し デジタルカメラの品質向上と開発期間短縮を実現

株式会社リコー

リコーは、デジタルカメラ製品のさまざまな機能を担う組み込みソフトウェアの品質を向上させ、複数の機種を効率的に開発することを目的として、ソフトウェアの構造を整理するという活動に取り組んでいる。そのツールとして同社が採用したのが、コードの構造を可視化し、改修の影響範囲を把握できる「Lattix」だった。

株式会社リコー

設立年月日	1936年2月6日
資本金	1,353億円(2010年3月31日現在)
代表者	代表取締役会長執行役員 桜井正光氏 代表取締役社長執行役員 近藤史朗氏
従業員数	108,500名(2010年3月31日現在・連結)
本社所在地	東京都中央区銀座8-13-1 リコービル
事業内容	複写機、印刷機、ファクシミリなど画像ソリューション分野、PC・サーバー、ネットワーク機器などのシステムソリューション分野、光学機器、半導体などの産業分野、デジタルカメラの製造・販売 ほか

品質向上のための 取り組みに着手

リコーは、デジタルコピー、ファクシミリ、プリンタなどのOA機器の分野において世界的に著名なメーカーだ。複写機や小型複合機では、国内でトップクラスのシェアを誇る。

そうしたOA機器とともに現在のリコー製品の柱となっているのが、デジタルカメラである。実はリコーは昭和初期からカメラを製造する老舗であり、1990年代まで35ミリフィルムのコンパクトカメラを中心に、高品質なマスマーケット向け製品を提供するメーカーとして広く認知されてきた。1995年に発売したデジタルカメラ「DC-1」以降は、有力なデジタルカメラメーカーとしての地位を確立。以降、フィルムカメラに引き継ぎ、コンパクトカメラ市場向けを中心に製品を

展開する。2009年12月には、レンズ、撮像素子、画像処理エンジンを交換可能なユニットに一体化した「GXR」を発売し、デジタルカメラの新しいスタイルとして注目されている。

そうしたリコーの高機能なデジタルカメラを支えているのが、組み込みソフトウェアだ。自動車、携帯電話、デジタル家電など電子制御で動くあらゆる機械・機器には、その心臓部にコンピュータシステムが組み込まれている。そのシステムにさまざまな機能を付加し、稼働させる役目を果たするのが組み込みソフトウェアだ。製品内部にあって容易に書き換えることができず、その出来が製品の品質を左右するため、どのメーカーも組み込みソフトウェアの品質管理には細心の注意を払っている。もちろん、リコーのデジタルカメラも例外ではない。

しかし、複数の機種を同時に効率的に開発するためには、より厳しい品質管理体制が求められる。そこでリコーが取り組んだのが、組み込みソフトウェアの構造を整理するという活動だった。

「当社では、全般的な組み込みソフトウェアの品質向上を実現するために、コードの構造を整理していくという活動に取り組んでいます。たとえば、変更箇所を特定しやすくするには、依存関係を整理しておかないといけません。そういう活動の積み重ねによって整

理されたものがあつたことで、たとえばGXRのようなユニット交換式カメラの開発も可能となりました」(リコー パーソナルマルチメディアカンパニー ICS設計室 牧 隆史氏)

依存関係の可視化を 実現

リコーのデジタルカメラ開発部門が、組み込みソフトの品質の観点で依存関係に着目しはじめたのは、2006年前後のことだという。

「それ以前から、静的解析などいくつかの品質向上のための取り組みは行っていましたが、依存関係の確認を組織的に行うことはあまりされてきませんでした。特に以前は、デジタルカメラに求められる機能が現在ほどは多くなかったため、開発者の人数も限られていたので、ある意味開発者に任されている部分があつたともいえます」

しかし、こうした属人的な品質管理では、さらなる開発効率の向上、開発期間の短縮という課題を解決することは難しい。それを改革する手段として、牧氏が中心となって取り組んだのが、組み込みソフトウェアのソースコードの依存関係を可視化することだった。

「これまでもソースコードを静的解析し



株式会社リコー
パーソナルマルチメディアカンパニー
ICS設計室 設計3G シニアスペシャリスト
牧 隆史氏

たり、アーキテクチャを図式化したりしてモジュール構造の依存関係を把握しようとしてきましたが、それを基に実際にコーディングする際に、ルールが守られているかどうかを確認する手段はなく、開発者に委ねられていたわけです。そこで、コードの依存関係を把握したり、呼び出し関係を解析したりできるツールを探したところ、ちょうどよいタイミングで私たちが探していた機能を持つLattixに出あったわけです」

開発効率と 開発スピードが向上

依存関係を把握したいという問題意識を持っていたリコーにとって、Lattixが提供する機能そのものが成果物と言えるものだった。牧氏はさっそく、Lattixの国内販売元であるテクマトリックスにコンタクトをとり、デモライセンスを入手して製品の評価を実施。有用性を確認した後に導入したという。

「組み込みソフトウェアのソースコードをLattixにすべて読み込ませ、アーキテクチャを検証するために利用しています。たとえば、新製品には新機能を実現するための新しいモジュールが入っています。新しいモジュールは、既存のモジュールから呼ばれたり、既存のモジュールを呼んだりという依存関係が必ず出てくるので、Lattixを使って依存関係が正しく実装されているかどうかを確認しています」

こうしたコーディング成果物の品質チェックは、プロジェクトごとに行われているという。

「かつてはソースコードを目で見て確かめていましたが、開発規模が大きくなるにつれ、すべてを見ることはできなくなってきました。明示的な依存関係は検出できても、そうでないところの検出は困難です。そのためにもツールの力は必要です」

牧氏は、Lattixを導入した効果も実感しているという。

「ルール違反になっている部分が可視化できる点が一番大きなメリットです。上級のプロジェクトマネージャにも、どのドメインに問題があるか一目で理解してもらえるので、対策が取り易くなりました。もちろん、このツールを使うことで製品開発における



Lattixを活用して開発された組み込みソフトウェアを搭載するリコーの最新デジタルカメラ「GXR」

問題の全てが解決するわけではありませんが、開発効率が向上したことで、結果的に品質が向上したと考えています」

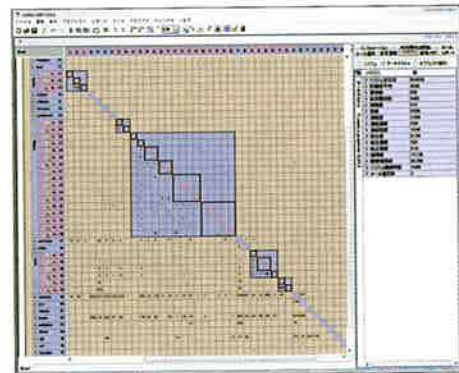
開発スピードも早くなったと評価する。

「開発スピードの向上にも寄与していると思います。正確に計測したわけではありませんが、ソースコードを目で見て確認しなければならない場合は、開発に着手する前に依存関係などを調べなければならず、それが時間のロスにつながっています。そういう部分がかなり短縮できました」

他のツールとの 連携に期待

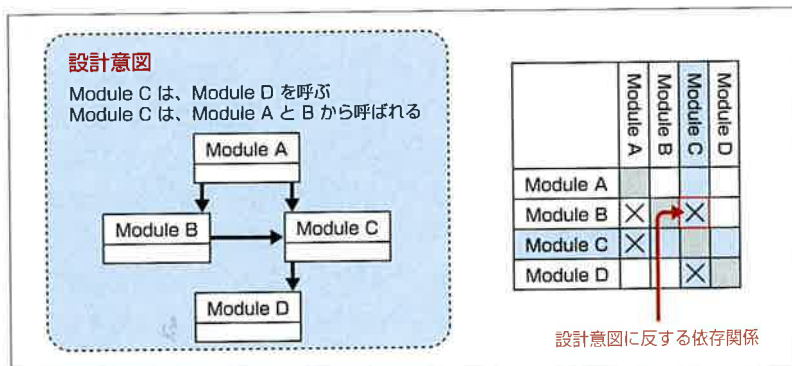
今後は、Lattixの進化に期待しているという。

「マトリックスで表示されるLattixの画面は、定量的で分かりやすいです。アーキテクチャ分析機能のベースとなるDSM



LattixのDSM (イメージ)

(Dependency Structure Matrix)が完成された手法なので、それを見るツールという観点では完成されていると思います。ただし、スペックの低いマシンで利用する場合、大量のソースコードを読み込んでセルの数が増えると、やや速度が低下することがありました。構造上、仕方のないことだとは思いますが、次のバージョンでパフォーマンスが向上したと聞いていますので、最新バージョンを使用するのを楽しみにしています。また、現バージョンでは、Understand (ソースコード解析ツール)の結果をLattixに読み込んだり、LattixからUnderstandを起動したりすることができですが、このようなツール連携を他のツールとも実現してもらえると助かります」



Lattixによるアーキテクチャルール違反の可視化イメージ

総販売代理店

TechMatrix

テクマトリックス株式会社

システムエンジニアリング事業部
ソフトウェアエンジニアリング営業部
lattix-info@techmatrix.co.jp

〒108-8588
東京都港区高輪4-10-8 京急第7ビル
TEL. 03-5792-8606 FAX 03-5792-8706
http://www.techmatrix.co.jp/quality/lattix/

〒541-0054
大阪市中央区南本町2-6-12 サンマリオンNBFタワー6F
TEL. 06-6243-3801 FAX. 06-6243-3803