ＤＳＬによる要求獲得でスーパーアジャイル

第２回ＴＰＳ／アジャイル研究会
有限会社来栖川電算 山口陽平
2011年7月8日(金)

アジェンダ

- **自己紹介**
- **せっかくＴＰＳ／アジャイル研究会なので**
  - 巨大滑り台で考えるアジャイル
- **ＤＳＬによる要求獲得でスーパーアジャイル**
  - ドメイン固有言語（ＤＳＬ）
  - ユーザ言語で仕様を記述・整理する
  - スーパーアジャイルじゃない？

山口陽平

- **プログラミング言語・型理論の研究者**
  - 世界を美しく記述することを夢見る３２歳
  - 人を驚かせてなんぼ
    - NativeコードよりPure Javaコード
    - １日でHaskellを作る
    - ハードリアルタイムJava VM
    - 1000台以上のサーバで構成されるペタバイト級分散データベース
    - ＰＣ上で1000万クエリ／秒を達成するKVS
- **来栖川電算**
  - 名古屋工業大学発（2003年設立）
  - ソフトウェアの品質・生産性の向上
  - IPA未踏ソフト経験者（を多数輩出）

※あくまでもイメージです。
実物に髪の毛はありません。

巨大滑り台で考えるアジャイル

- 開発者「巨大滑り台できました！」
- お客様「あれ？左の端、…モアイ？」
- 開発者「ハイ！特にご希望がなかったので、今流行りのモアイにしておきました！」

巨大滑り台が示唆すること

作った後で気になることは多いが、作る前は何を気にしてよいか分からない。

⇒捉えどころがない困難

- 開発者「巨大滑り台できました！」
- お客様「あれ？左の端、…モアイ？」
- 開発者「ハイ！特にご希望がなかったので、流行りのモアイにしておきました！」

困難の根源１

そもそもお客様は表面的な要求しか把握していない、些細な要求・偽りの要求に囚われていることが多い。しかも世の中が変わって行くからややこしくなる。

- 開発者「石と湯呑み台できました！」
- お客様「あれ？左の端、…モアイ？」
- 開発者「ハイ！特にご希望がなかったので、流行りのモアイにしておきました！」

困難の根源２

お客様は都合のよい効果だけに夢中で、副次的効果の絡まりに関心や理解が乏しく、想像も苦手。システムの姿を劇的に変えてしまうかもしれないのに。

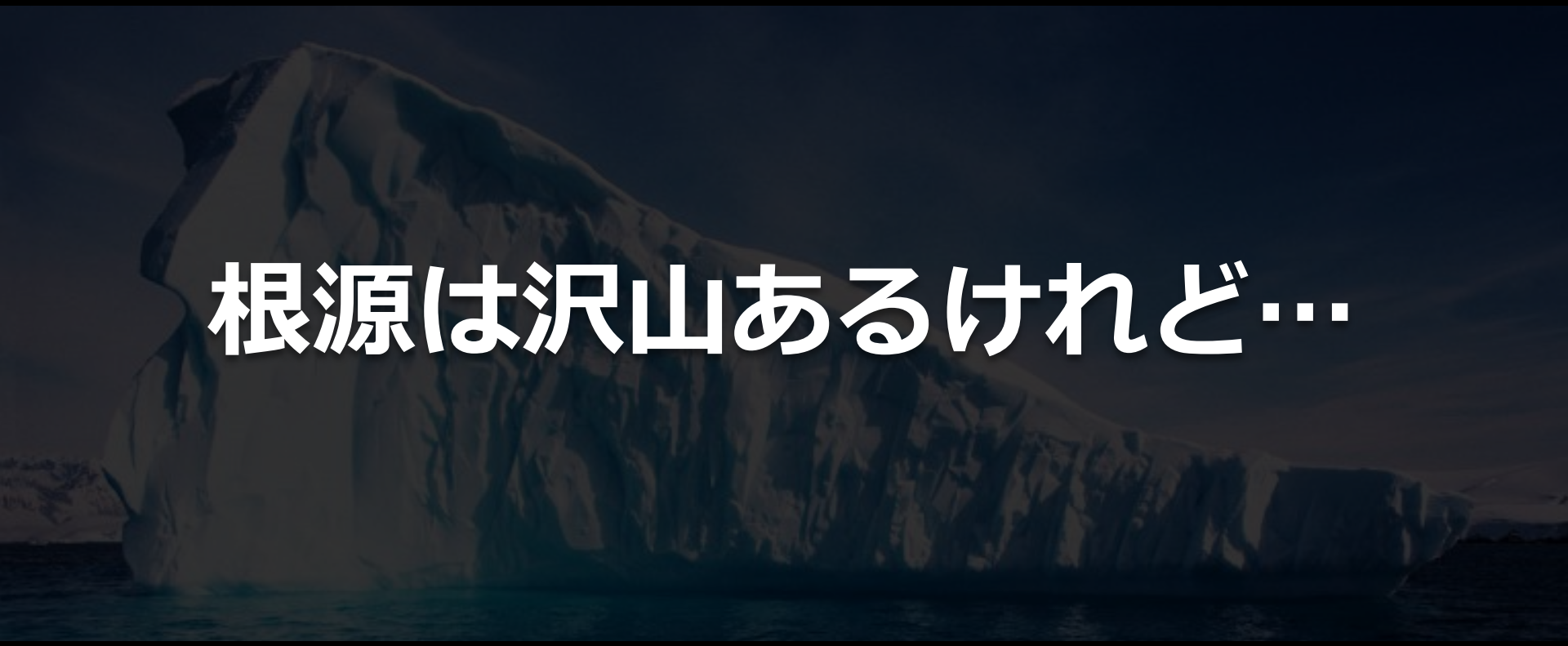
- 開発者「巨大滑り台できました！」
- お客様「あれ？左の端、…モアイ？」
- 開発者「ハイ！特にご希望がなかったので、流行りのモアイにしておきました！」

困難の根源３

要求を持っているのはお客様。気付いていない・もやもやした表現し難い要求を開発者に伝えなくてはならない。どうやって？

- 開発者「巨大滑り台できました！」
- お客様「あれ？左の端、…モアイ？」
- 開発者「ハイ！特にご希望がなかったので、流行りのモアイにしておきました！」

困難の根源９９９９



- 開発者「巨大滑り台できました！」
- お客様「あれ？左の端、…モアイ？」
- 開発者「ハイ！特にご希望がなかったので、流行りのモアイにしておきました！」

根源に人間系強化で挑むのが…

アジャイル

お客様と開発者からなる
チームの認知力・想像力・
伝達力を強化する補助魔法

- 開発者「巨大滑り台できました！」
- お客様「あれ？左の端、…モアイ？」
- 開発者「ハイ！特にご希望がなかったので、流行りのモアイにしておきました！」

補助魔法『アジャイル』の効能

1. **認知・想像・伝達の機会を作る**
   ストーリーの作成・バックログの作成・受け入れテスト・スプリントレビュー
2. **認知・想像・伝達の機会を増やす**
   開けた作業空間・反復・スプリント・頻繁な振り返り
3. **認知力・想像力・伝達力を増幅する**
   共通の用語・ストーリーカード・バックログ・動くソフトウェア・現物

- 開発者「巨大滑り台できました！」
- お客様「あれ？左の端、…モアイ？」
- 開発者「ハイ！特にご希望がなかったので、流行りのモアイにしておきました！」

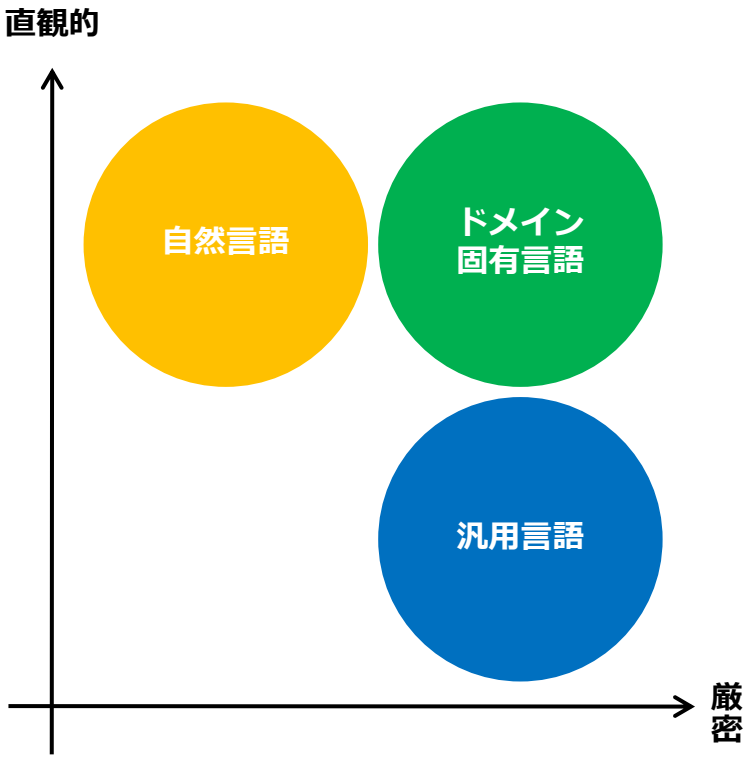
ドメイン固有言語（Domain Specific Language）

お客様の頭の中を整理し、彼らの想像力と表現力を高めれば、最良の要求や仕様を引き出せる。正確で厳密に記述された要求や仕様は機械的に処理できる。ドメイン固有言語はコミュニケーションやプログラミングを効率化する強力なツールとなる。

ドメイン固有言語

そもそもドメイン固有言語とは？

- **特定の知識を表現・解釈するための形式的手法**
  - **自然言語**より伝えやすい。
    - 曖昧で混乱を誘う表現・解釈が少ない。
    - 整合性・効率性・過不足を検証しやすく、場合によっては実行できる。
  - **汎用言語**より分かりやすい。
    - ドメインの知識を直接的かつ簡潔に表現できる。
    - ドメインの専門家が理解でき、場合によっては記述もできる。



ドメイン固有言語

身近な例

- **文法定義**
  - LEX, YACC, ANTLR, Javacc, TMF, MPS
- **文書作成**
  - sed, TeX, HTML, Wiki記法
- **グラフ作成**
  - gnuplot, GraphViz
- **ファイル処理**
  - UNIX shell, find, grep, make, ANT
- **データ処理**
  - SQL, XML, XSLT, XPath, XQuery

ドメイン固有言語

なぜ重要なのか？

背景

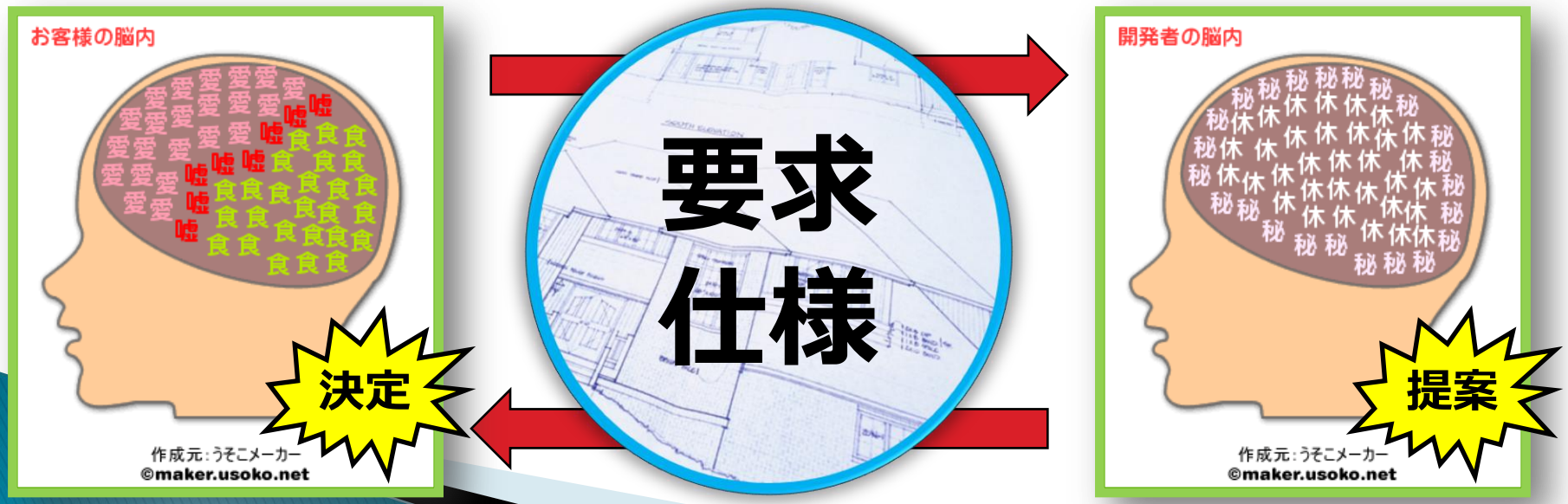
- 根本的な困難
- 技術の変化
- 社会の変化

効果

- コミュニケーション
  - 想像をガイドする
  - 意思疎通をスムーズにする
  - 判断を合理的にする
- プログラミング
  - 確認が楽になる
  - 修正が楽になる
  - 専門による分業ができる
  - 記述が資産になる

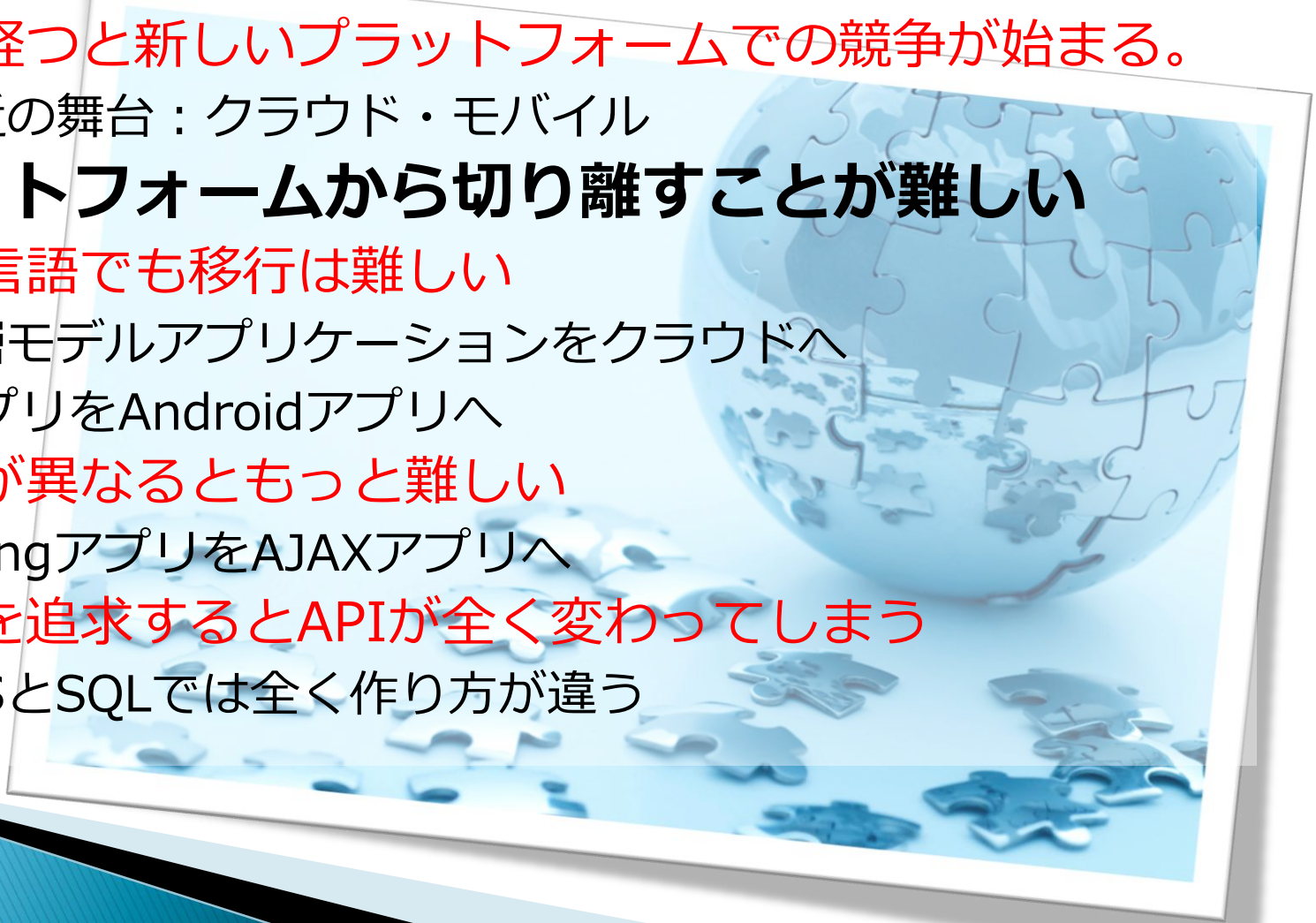
根本的な困難

- **２種類の専門家が知識を交換する**
  - お客様はシステムが何を為すべきかを決定できる。
  - 開発者はシステムをどのように実現すべきかを提案できる。
- **想像や表現が難しい知識を扱う**
  - 暗黙的・非直観的・抽象的でたくさんの知識を厳密に整理・検証・伝達し合わなければならない。



技術の変化

- **プラットフォームが安定しない**
  - 数年経つと新しいプラットフォームでの競争が始まる。
    - 最近の舞台：クラウド・モバイル
- **プラットフォームから切り離すことが難しい**
  - 同じ言語でも移行は難しい
    - 三層モデルアプリケーションをクラウドへ
    - iアプリをAndroidアプリへ
  - 言語が異なるともっと難しい
    - SwingアプリをAJAXアプリへ
  - 性能を追求するとAPIが全く変わってしまう
    - KVSとSQLでは全く作り方が違う

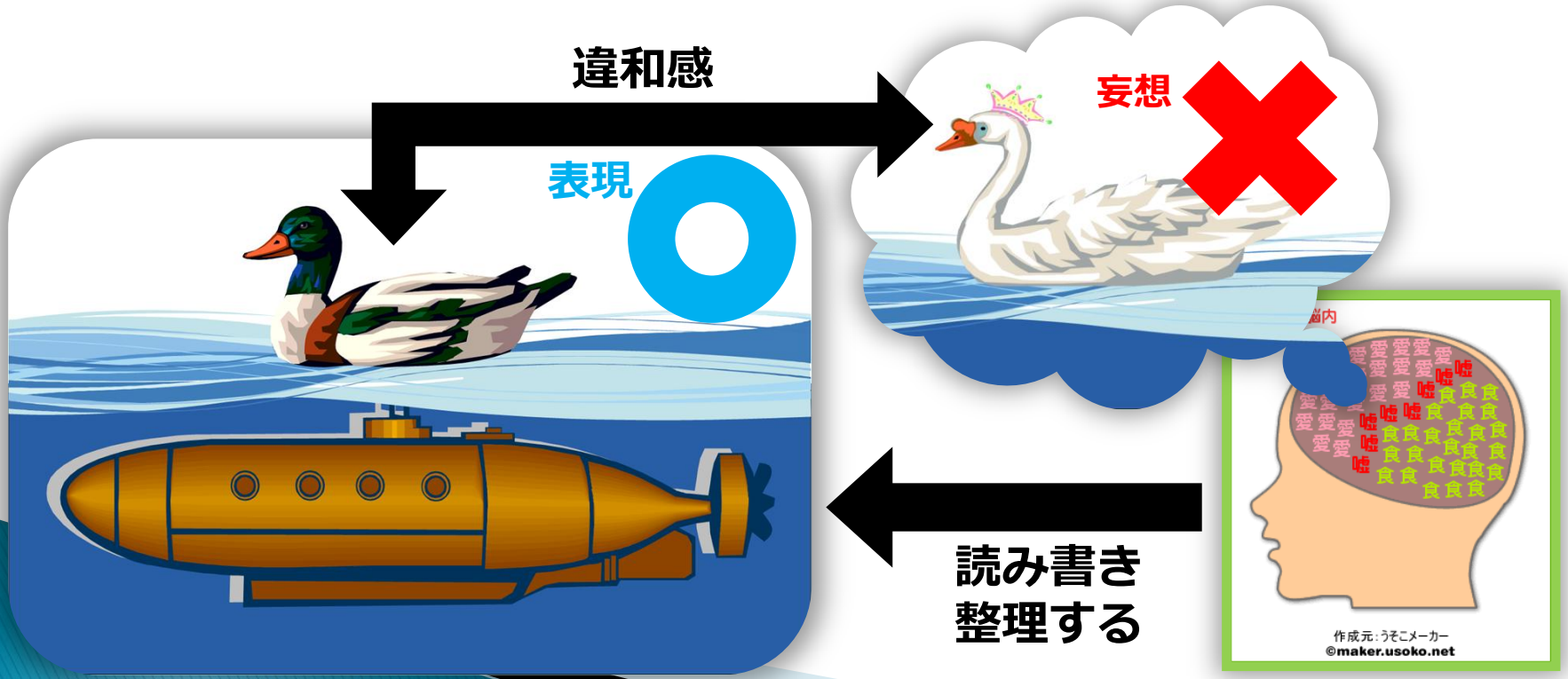


社会の変化

- **お客様がソフトウェアを内製する傾向にある**
  - 数は少ないが企業内アマチュアプログラマが増えている。
    - Excel・Access・VBA・VB・SQL・PHP・Apache・Unix
  - 変化へ素早く対応することが重視される。
    - 入出力項目・ワークフローの絶え間ない改善
  - セキュリティが重視される。
    - 企業ノウハウ・個人情報の流出阻止
- **プロがやるべき仕事が変わってきている**
  - アマチュアプログラマでは不可能な規模・複雑さの開発
  - アマチュアプログラマでも使えるツールの提供

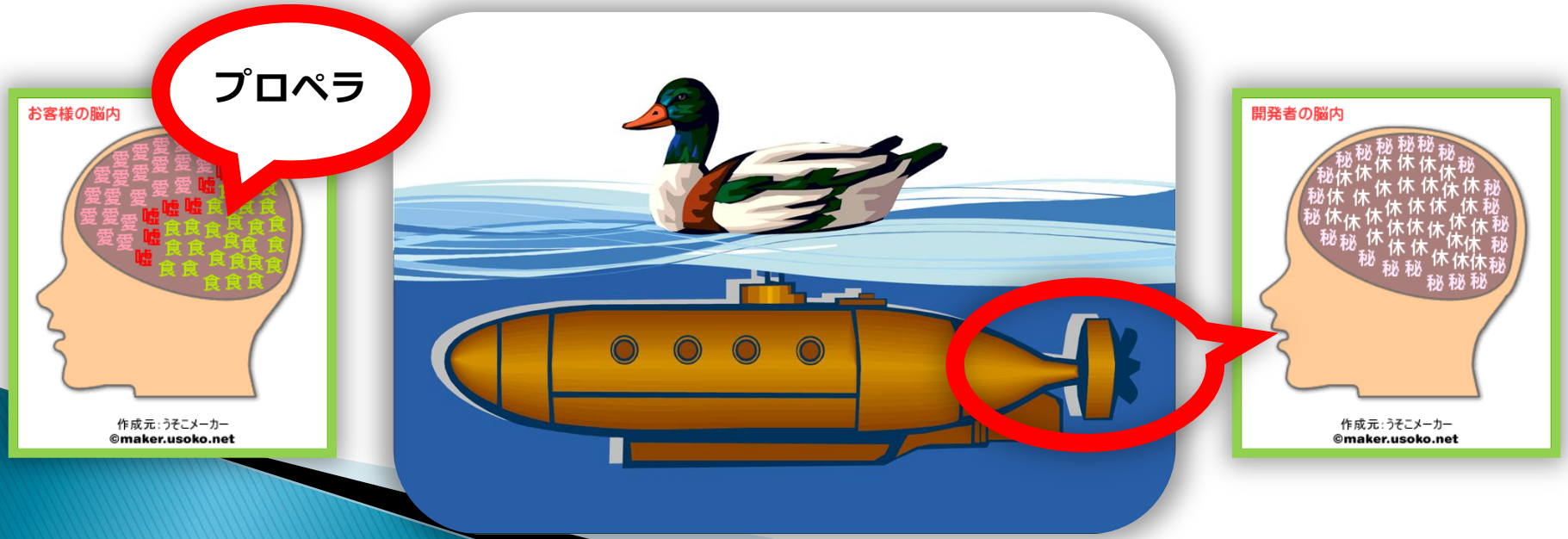
想像をガイドする

- **特徴を捉えた的確な表現方法が知識の姿形となる。**
  - 直観で認識できる世界が広がり、勘が良くなる。
- **形式的な解釈方法が様々な気付きをもたらす。**
  - 非直観的・暗黙的な知識をあぶり出す。



意思疎通をスムーズにする

- **特徴を捉えた的確な表現方法が会話を助ける。**
  - 注目すべき点が明確になる。
  - 混同や飛躍がなくなる。
- **共有された解釈方法が曖昧さを減らす。**
  - 誰がいつ解釈しても同じになる。

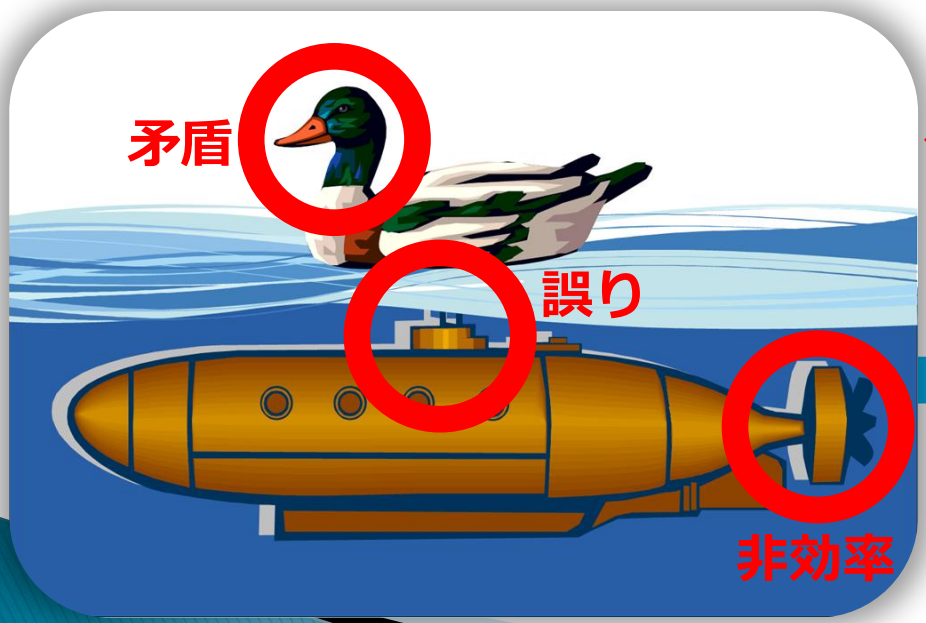


判断を合理的にする

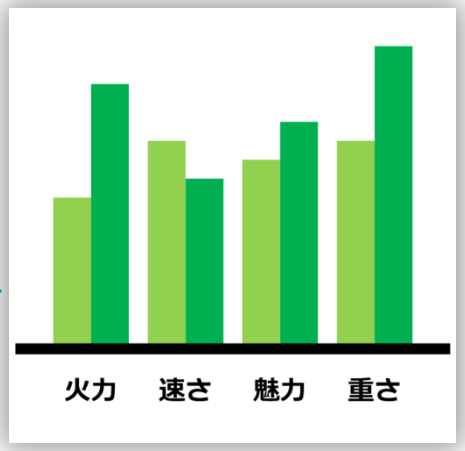
- **形式的な解釈方法が判断材料を増やす。**
  - 思いもよらない・予想に反する事柄の発見が常識や偏見にとらわれない判断を可能にする。
- **記述が地図となり見通しを良くする。**
  - 頭の中に入りきらないほど大規模で複雑な知識を扱える。

確認が楽になる

- **機械的な解釈をツールで自動化できる。**
  - 複雑な解釈方法を伴うドメインも扱える。
  - 矛盾や誤りや非効率を自動的に発見できる。
  - 確認時間が短縮されるため思考錯誤しやすくなる。



ツール



レポート

修正が楽になる

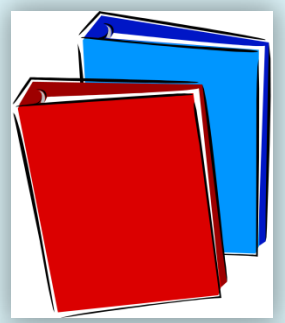
- **言語処理系**は保守性を高める
  - プログラムやドキュメントを生成する**コンパイラやインタプリタ**は関連する記述の一貫性を自動的に維持する。
  - 拡散する記述を閉じ込めるため、設計が美しくなる。
  - 最適化により性能が高まる。

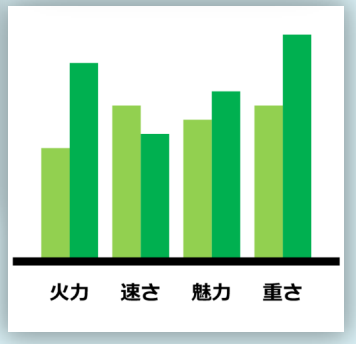
一貫性維持

IDENTIFIER	COMPONENT	VALUE
家族	Panel	
家族.名前	TextBox	
家族.平均身長	NumberBox	average(身長 from 構成員s)
家族.平均体重	NumberBox	average(体重 from 構成員s)
家族.平均BMI	NumberBox	average(BMI from 構成員s)
家族.構成員s	Table	
家族.構成員s[i]	Row	
家族.構成員s[i].名前	TextBox	
家族.構成員s[i].身長	NumberBox	
家族.構成員s[i].体重	NumberBox	
家族.構成員s[i].BMI	NumberBox	体重 / (身長 * 身長)

拡散防止
性能向上

言語処理系





ユーザ言語で仕様を記述・整理する »

生産記録システムの事例から言語指向プログラミングを理解する。

ユーザ言語で仕様を記述・整理する

生産記録システム

撹拌作業記録票

溶剤生産課

2010 年 8 月 3 日

作業者名	作業開始時刻	～	作業終了時刻
天海	8：15	～	8：45
如月	9：10	～	10：10
萩原	10：15	～	10：30
高槻	11：00	～	12：00
秋月	13：05	～	13：25
三浦	14：00	～	16：00
菊地	16：00	～	16：05
水瀬	16：05	～	16：06
双海	16：06	～	17：00
	：	～	：
	：	～	：

開始温度	218 ℃
終了温度	225 ℃
☑ 回転式	245～255
回転数	215 rpm
流量	42 l/min
☑ 除去記録	

確認印	
課長	検査員

作業者マスタから、選択式

別の所でが

- 要求
  - 記録用紙を電子化したい。
  - 記録用紙
    - 頻繁に様式が変化する。
      - 定期的な改善活動がある。
    - 種類が多い。
      - 商品ごとの工程数に比例する。
    - 記入欄が複雑かつ多い。
      - 記入内容やその正しさが他の（記録用紙の）記入内容や外部情報に依存して決まることがある。
      - 一度に全ての記入欄が埋まらないことがある。
      - 条件によって記入方法が変化することがある。

ユーザ言語で仕様を記述・整理する

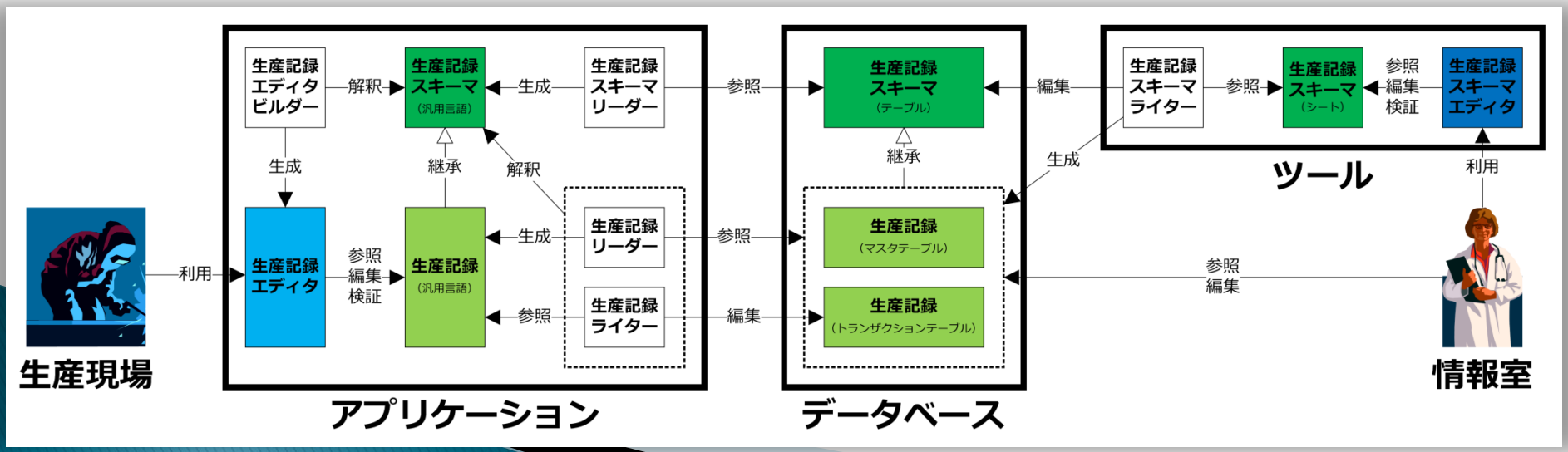
お客様の体制

- **生産現場（数百数人）**
  - 記録用紙を使って業務を行っている。
    - 記入欄が何のためにどのように使われているか知っている。
  - ITリテラシーがある人がちらほらいる。
    - ExcelやWordを使える人がいるだけでなく、稀にアクセスでプログラムできる人もいる。
- **情報室（十数人）**
  - 社内システムのデータベースを管理している。
    - 生産現場から収集した情報をデータベースへ登録したり、データベースから抽出した情報を生産現場へ提供している。
  - ITリテラシーが高い。
    - 全ての人がSQL・VB・Access・Excelを使える。

ユーザ言語で仕様を記述・整理する

提案した仕組

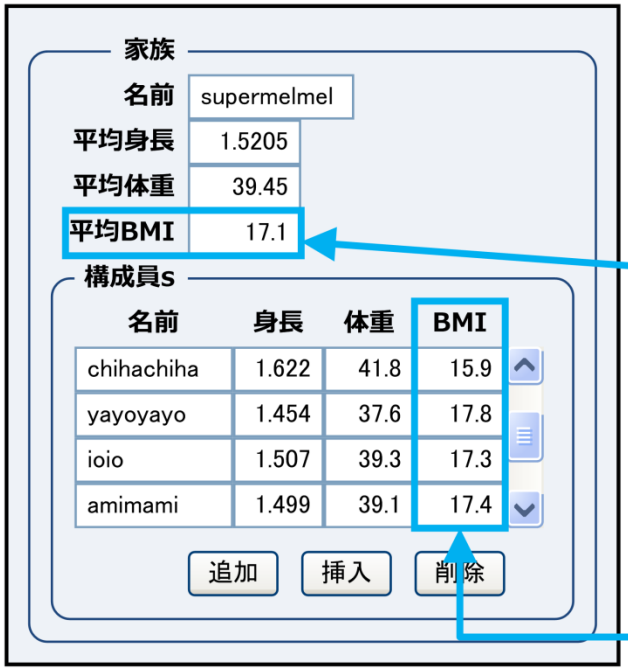
- **ExcelとSQLでカスタマイズできるシステム**
  - **様式（生産記録スキーマ）**をExcelで編集できる。
  - **記録用紙（生産記録エディタ）**をすぐに確認できる。
  - 記入内容を格納する**生産記録**（トランザクションテーブル）と初期値や選択肢などを格納する**生産記録**（マスタテーブル）を生成できる。後者をトリガで更新するかビューにすることで他システムのマスタテーブルと連動できる。



ユーザ言語で仕様を記述・整理する

生産記録スキーマ言語

- **対応が分かりやすく編集しやすい表現**
  - 1個の記入欄を1行で表す。
    - 階層的識別子を使う。特徴を各列で表す。
  - 識別子を使った関係式で特徴を宣言的に表す。



IDENTIFIER	COMPONENT	VALUE
家族	Panel	
家族.名前	TextBox	
家族.平均身長	NumberBox	average(身長 from 構成員s)
家族.平均体重	NumberBox	average(体重 from 構成員s)
家族.平均BMI	NumberBox	average(BMI from 構成員s)
家族.構成員s	Table	
家族.構成員s[i]	Row	
家族.構成員s[i].名前	TextBox	
家族.構成員s[i].身長	NumberBox	
家族.構成員s[i].体重	NumberBox	
家族.構成員s[i].BMI	NumberBox	体重 / (身長 * 身長)

ユーザ言語で仕様を記述・整理する

生産記録スキーマ言語の変遷

- **ユーザ言語は主に次のフェーズを経て獲得できる**
  - 抽出フェーズ
    - お客様も気づかない隠された要求や仕様を洗い出す。
  - 整理フェーズ
    - 全ての要求や仕様を正確に表現し、お客様と合意する。
  - 自動化フェーズ
    - 要求や仕様を実行する。
- **続くスライドでは生産記録スキーマ言語をどのように獲得したかを上記のフェーズごとに紹介する**

抽出フェーズのスキーマ言語

- **お客様も気付かない隠された要求や仕様を洗い出すことが目的**
  - 記録用紙から言語とスキーマを抽出する。記録用紙とスキーマをお客様に見せながら対応を説明する。
    - 解釈が間違っていると教えてくれる。
  - 各項目はヒアリングの項目になる。
    - 具体的に1つ1つ確認すると分かってくれる。
    - 複数人でするため書き方の合意はとる。
  - 合意した書き方で書けなくても書く。
    - マークしておき、後で言語に反映する。
    - 書けないことは隠された要求の発見に繋がりやすい。

項目
識別子
型
制約
空白不可
値
コンポーネント
フォーマット
単位
表示
有効
マスタ記録
トランザクション記録
備考

抽出フェーズのスキーマ

識別子	型	範囲	空白不可	値	コンポーネント	フォーマット	単位	表示	有効	マスタ記録	トランザクション記録	備考
記録票	Row				Folding						○	
記録票.担当者s	Collection	1..								○		
記録票.担当者s[i]	Row							false		△		
記録票.担当者s[i].ID	String	0..16								○		
記録票.担当者s[i].名前	String	0..64						false		○		
記録票.担当者s[i].toString	String	0..81		ID + ":" + 名前								
記録票.作業s	Collection	1..			Table						○	
記録票.作業s[i]	Row				Flow						△	
記録票.作業s[i].担当者	Row		○		DropDownList	担当者s					△	
記録票.作業s[i].担当者.ID	String	0..16			NumberLabel						○	
記録票.作業s[i].担当者.名前	String	0..64			StringLabel						○	
記録票.作業s[i].担当者.toString	String	0..81		ID + ":" + 名前	StringLabel							
記録票.作業s[i].開始時刻	DateTime		○		DateTime	"YYYY/MM/DD hh:mm"					○	
記録票.作業s[i].終了時刻	DateTime		○		DateTime	"YYYY/MM/DD hh:mm"					○	
記録票.温度	Row				Flow						△	
記録票.温度.開始	Integer16	270..290	○		Number	"##0"	"℃"				○	
記録票.温度.終了	Integer16	270..290	○		Number	"##0"	"℃"				○	
記録票.回転式	Boolean	true	○		Check						○	
記録票.回転数	Integer16	245..255	○		Number	"##0"	"rpm"		回転式		○	
記録票.流量	Integer16	流量下限値..流量上限値	○		Number	"##0"	"l/min"				○	
記録票.流量下限値	Integer16		○	回転式 ? 30S : 80S		"##0"	"l/min"	false				
記録票.流量上限値	Integer16		○	回転式 ? 50S : 100S		"##0"	"l/min"	false				
記録票.除去記録	Boolean		○		Check						○	

整理フェーズのスキーマ言語

- **全ての要求や仕様を正確に表現し、お客様と合意することが目的**
  - 洗い出したスキーマを正確に表現できるより単純な言語を探し、それでスキーマを再表現する。
    - 言語処理系が作りやすくなる。
    - 開発コスト・検証力・性能・表現力のバランスが難しい。
  - この時期から編集や検証を自動化するツールを作れるようになる。
    - 言語が成長している間は、使い捨てツールと手作業を組み合わせて編集するとよい。

表現が変わった項目
識別子

増えた項目
表示識別子

整理フェーズのスキーマ

類似語統一

英語名生成

共通部分抽出

識別子	型	範囲	空白	値	表示識別子	コンポーネント	フォーマット	単位	表示	有効	マスタ記録	トランザクション記録	備考
Header	Row				ヘッダ	Folding					○	○	
Header.Workers	Collection	1..			ヘッダ.担当者s						○		
Header.Workers[i]	Row				ヘッダ.担当者s[i]				false		△		
Header.Workers[i].Id	String	0..16			ヘッダ.担当者s[i].ID						○		
Header.Workers[i].Name	String	0..64			ヘッダ.担当者s[i].名前				false		○		
Header.Workers[i].toString	String	0..81		Id + ":" + Name	ヘッダ.担当者s[i].toString								
Report	Row				記録票	Folding						○	
Report.Tasks	Collection	1..			記録票.作業s	Table						○	
Report.Tasks[i]	Row				記録票.作業s[i]	Flow						△	
Report.Tasks[i].Worker	Row		○		記録票.作業s[i].担当者	DropDownList	Header.Workers					△	
Report.Tasks[i].Worker.Id	String	0..16			記録票.作業s[i].担当者.ID	NumberLabel						○	
Report.Tasks[i].Worker.Name	String	0..64			記録票.作業s[i].担当者.名前	StringLabel						○	
Report.Tasks[i].Worker.toString	String	0..81		Id + ":" + Name	記録票.作業s[i].担当者.toString	StringLabel							
Report.Tasks[i].StartTime	DateTime		○		記録票.作業s[i].開始時刻	DateTime	"YYYY/MM/DD hh:mm"					○	
Report.Tasks[i].EndTime	DateTime		○		記録票.作業s[i].終了時刻	DateTime	"YYYY/MM/DD hh:mm"					○	
Report.Temperature	Row				記録票.温度	Flow						△	
Report.Temperature.Start	Integer16	270..290	○		記録票.温度.開始	Number	"##0"	"℃"				○	
Report.Temperature.End	Integer16	270..290	○		記録票.温度.終了	Number	"##0"	"℃"				○	
Report.IsRolling	Boolean	true	○		記録票.回転式	Check						○	
Report.RollingCount	Integer16	245..255	○		記録票.回転数	Number	"##0"	"rpm"		IsRolling		○	
Report.Flow	Integer16	FlowLower..FlowUpper	○		記録票.流量	Number	"##0"	"l/min"				○	
Report.FlowLower	Integer16		○	IsRolling ? 30S : 80S	記録票.流量下限値		"##0"	"l/min"	false				
Report.FlowUpper	Integer16		○	IsRolling ? 50S : 100S	記録票.流量上限値		"##0"	"l/min"	false				
Report.Cleaned	Boolean		○		記録票.除去記録	Check						○	

自動化フェーズのスキーマ言語

- **要求や仕様を実行することが目的**
  - 開発者やアプリケーションや実行環境の都合で必要な情報を表現できるように言語を拡張し、スキーマにその情報を付加する。
    - 多くの情報（のデフォルト値）は既存の情報から補えるので、自動編集ツールを作るとよい。
  - お客様が自分でスキーマを修正し、他システムと連携できるようにサポートする。
    - 繰り返し作業やよくある間違いを言語やツールの改善で抑制する。

減った項目
範囲
空白不可
フォーマット
単位

増えた項目
型パラメータ
制約
コンポーネントパラメータ
マスタ実態名
トランザクション実態名

ユーザ言語で仕様を記述・整理する ＞ 生産記録スキーマ言語の変遷

自動化フェーズのスキーマ

識別子	型	型パラメータ	制約	値	表示識別子	コンポーネント	コンポーネントパラメータ	表示	有効	マスタ記録	トランザクション記録	マスタ実態名	トランザクション実態名	備考	検証結果
Header	Row				ヘッダ	Folding				○	○	M_Header	T_Header		
Header.Workers	Collection	1			ヘッダ.担当者s					○		M_Header_Workers			
Header.Workers[i]	Row				ヘッダ.担当者s[i]			false		△		M_Header_Workers			
Header.Workers[i].Id	String	0, 16			ヘッダ.担当者s[i].ID					○		Id			
Header.Workers[i].Name	String	0, 64			ヘッダ.担当者s[i].名前			false		○		Name			
Header.Workers[i].toString	String	0, 81		Id + ":" + Name	ヘッダ.担当者s[i].toString										
Report	Row				記録票	Folding					○		T_Report		
Report.Tasks	Collection	1			記録票.作業s	Table					○		T_Report_Tasks		
Report.Tasks[i]	Row				記録票.作業s[i]	Flow					○		T_Report_Tasks		
Report.Tasks[i].Worker	Row		NotNull(value.Id)		記録票.作業s[i].担当者	DropDownList	Header.Workers				△		T_Report_Tasks		
Report.Tasks[i].Worker.Id	String	0, 16			記録票.作業s[i].担当者.ID	NumberLabel					○		Worker_Id		
Report.Tasks[i].Worker.Name	String	0, 64			記録票.作業s[i].担当者.名前	StringLabel					○		Worker_Name		
Report.Tasks[i].Worker.toString	String	0, 81		Id + ":" + Name	記録票.作業s[i].担当者.toString	StringLabel									
Report.Tasks[i].StartTime	DateTime		NotNull(value)		記録票.作業s[i].開始時刻	DateTime	YYYY/MM/DD hh:mm"				○		StartTime		
Report.Tasks[i].EndTime	DateTime		NotNull(value)		記録票.作業s[i].終了時刻	DateTime	YYYY/MM/DD hh:mm"				○		EndTime		
Report.Temperature	Row				記録票.温度	Flow					△		T_Report		
Report.Temperature.Start	Integer16		NotNull(value) + Between(value, 270, 290)		記録票.温度.開始	Number	##0", "℃"				○		Temperature_Start		
Report.Temperature.End	Integer16		NotNull(value) + Between(value, 270, 290)		記録票.温度.終了	Number	##0", "℃"				○		Temperature_End		
Report.IsRolling	Boolean		NotNull(value) + Equal(value, true)		記録票.回転式	Check					○		IsRolling		
Report.RollingCount	Integer16		NotNull(value) + Between(value, 245, 255)		記録票.回転数	Number	##0", "rpm"		IsRolling		○		RollingCount		
Report.Flow	Integer16		NotNull(value) + Between(value, FlowLower, FlowUpper)		記録票.流量	Number	##0", "l/min"				○		Flow		
Report.FlowLower	Integer16		NotNull(value)	IsRolling ? 30S : 80S	記録票.流量下限値		##0", "l/min"	false							
Report.FlowUpper	Integer16		NotNull(value)	IsRolling ? 50S : 100S	記録票.流量上限値		##0", "l/min"	false							
Report.Cleaned	Boolean		NotNull(value)		記録票.除去記録	Check					○		Cleaned		

新表現

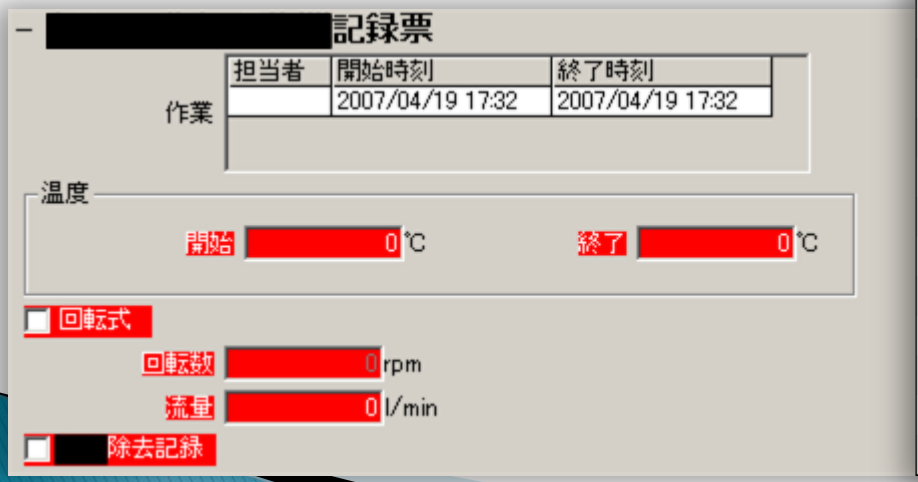
新表現

生成

生産記録スキーマからの生成物

- **フォーム＆テーブルスキーマ**
  - 選択肢はマスタから読み込み
  - 入力項目：3n + 9
  - 検査項目：3n + 13
  - 計算項目：n + 3

共通仕様	6行
個別仕様	18行

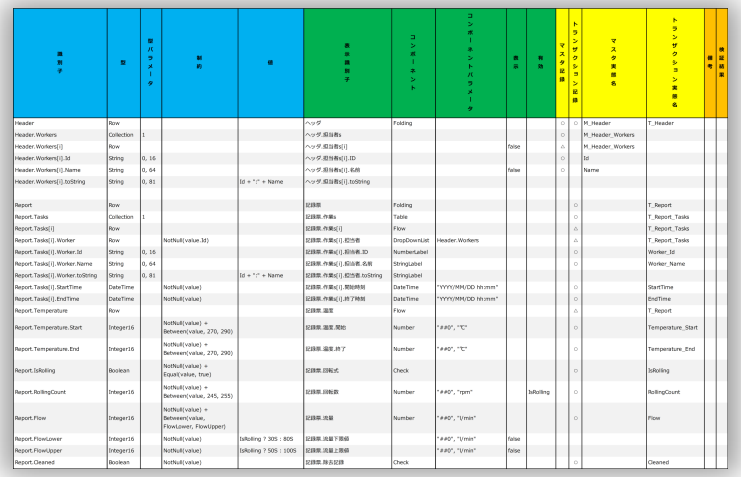


テーブル	フィールド	型	キー
M_Header	LotId	varchar(16)	○
M_Header_Workers	LotId	varchar(16)	○
M_Header_Workers	Id	varchar(64)	○
M_Header_Workers	Name	varchar(64)	
T_Header	LotId	varchar(16)	○
T_Report	LotId	varchar(16)	○
T_Report	Temperature_Start	short	
T_Report	Temperature_End	short	
T_Report	IsRolling	boolean	
T_Report	RollingCount	short	
T_Report	Flow	short	
T_Report	Cleaned	boolean	
T_Report_Tasks	LotId	varchar(16)	○
T_Report_Tasks	i	varchar(16)	○
T_Report_Tasks	Worker_Id	varchar(64)	
T_Report_Tasks	Worker_Name	varchar(64)	
T_Report_Tasks	StartTime	datetime	
T_Report_Tasks	EndTime	datetime	

ユーザ言語で仕様を記述・整理する

まとめ

- **うまくいったポイント**
  - お客様の能力に合った言語仕様
  - 記録用紙と対応がとりやすい言語仕様
  - 強力な検証機能を持つ言語仕様
  - 言語仕様からプラットフォームの知識を排除
  - 言語仕様の変化を考慮
- **効果**
  - お客様による試行錯誤
  - 短期間に大量の画面を作成
  - 他のプラットフォームへの乗換
  - ベンダーロック



識別子	型	個数パラメータ	制約	値	表示属性式子	コンポーネント	コンポーネントパラメータ	横並	改行	マスタ定義	トランザクションID定義	マスタ実態名	トランザクション実態名	補足説明	補足説明
Header	Row				ヘッダ	Folding				○	○	M_Header	T_Header		
Header.Workers	Collection	1			ヘッダ.担当者s					○		M_Header_Workers			
Header.Workers[i]	Row				ヘッダ.担当者s[i]			false		△		M_Header_Workers			
Header.Workers[i].Id	String	0, 16			ヘッダ.担当者s[i].ID					○		Id			
Header.Workers[i].Name	String	0, 64			ヘッダ.担当者s[i].名前			false		○		Name			
Header.Workers[i].toString	String	0, 81		Id + ":" + Name	ヘッダ.担当者s[i].toString										
Report	Row				記録票	Folding				○			T_Report		
Report.Tasks	Collection	1			記録票.作業s	Table				○			T_Report_Tasks		
Report.Tasks[i]	Row				記録票.作業s[i]	Flow				△			T_Report_Tasks		
Report.Tasks[i].Worker	Row		NotNull(value.Id)		記録票.作業s[i].担当者	DropDownList	Header.Workers			△			T_Report_Tasks		
Report.Tasks[i].Worker.Id	String	0, 16			記録票.作業s[i].担当者.ID	NumberLabel				○			Worker_Id		
Report.Tasks[i].Worker.Name	String	0, 64			記録票.作業s[i].担当者.名前	StringLabel				○			Worker_Name		
Report.Tasks[i].Worker.toString	String	0, 81		Id + ":" + Name	記録票.作業s[i].担当者.toString	StringLabel									
Report.Tasks[i].StartTime	DateTime		NotNull(value)		記録票.作業s[i].開始時刻	DateTime	"YYYY/MM/DD hh:mm"			○			StartTime		
Report.Tasks[i].EndTime	DateTime		NotNull(value)		記録票.作業s[i].終了時刻	DateTime	"YYYY/MM/DD hh:mm"			○			EndTime		
Report.Temperature	Row				記録票.温度	Flow				△			T_Report		
Report.Temperature.Start	Integer16		NotNull(value) + Between(value, 270, 290)		記録票.温度.開始	Number	"#0", "℃"			○			Temperature_Start		
Report.Temperature.End	Integer16		NotNull(value) + Between(value, 270, 290)		記録票.温度.終了	Number	"#0", "℃"			○			Temperature_End		
Report.IsRolling	Boolean		NotNull(value) + Equal(value, true)		記録票.回転式	Check				○			IsRolling		
Report.RollingCount	Integer16		NotNull(value) + Between(value, 245, 255)		記録票.回転数	Number	"#0", "rpm"		IsRolling	○			RollingCount		
Report.Flow	Integer16		NotNull(value) + Between(value, FlowLower, FlowUpper)		記録票.流量	Number	"#0", "l/min"			○			Flow		
Report.FlowLower	Integer16		NotNull(value)	IsRolling ? 305 : 805	記録票.流量下限値		"#0", "l/min"	false							
Report.FlowUpper	Integer16		NotNull(value)	IsRolling ? 505 : 1005	記録票.流量上限値		"#0", "l/min"	false							
Report.Cleaned	Boolean		NotNull(value)		記録票.除去記録	Check				○			Cleaned		

ユーザ言語で仕様を記述・整理する ＞ 効果

お客様による試行錯誤

- **PDCAサイクル高速化**
  - 発注に伴うオーバヘッドを気にしない。
    - 小間切れ・突発
  - 何が良いか判断できる人が納得いくまでする。
    - 品質が向上
    - 誇りと一体感が発生
  - 共通理解の深まり
    - お客様が賢くなり、コミュニケーションコストが減少

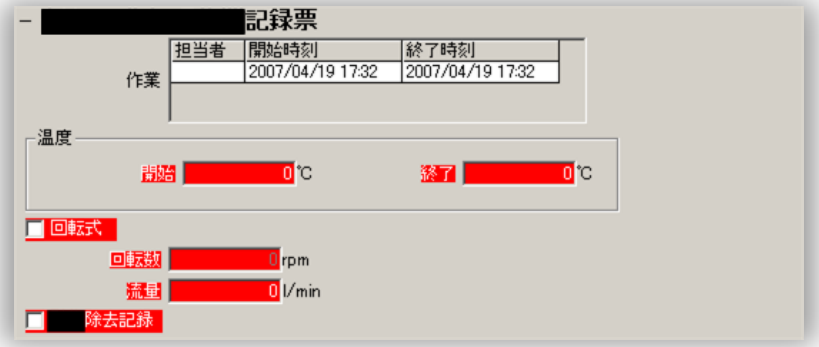
記録用紙1枚の電子化にかかる時間

タスク	従来手法	LOP手法
仕様書作成	4 時間	1 時間
内部設計書作成	4 時間	0 時間
プログラム作成	16 時間	0 時間
内部試験	12 時間	0 時間
データ作成	2 時間	2 時間
試験	2 時間	2 時間
合計	40 時間	5 時間

ユーザ言語で仕様を記述・整理する ＞ 効果

短期間に大量の画面作成

- **開発したもの**
  - Collectionライブラリ
  - SwingもどきGUIライブラリ
  - 言語処理系×10個
  - 記録用紙の生産記録スキーマ
  - 各種ドキュメント
- **工数**
  - 15人月
  - 1.5画面/人日
  - 驚異的な生産性

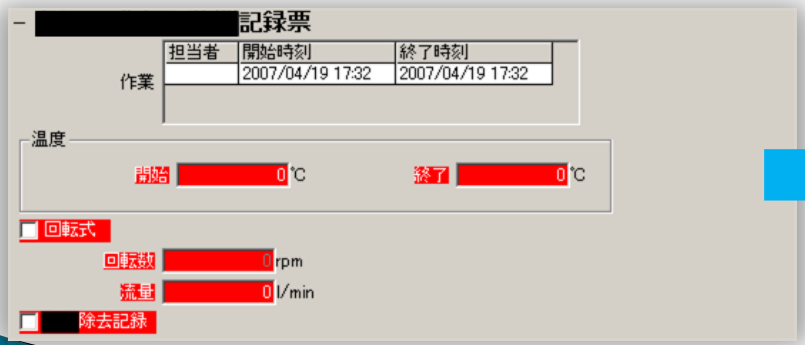


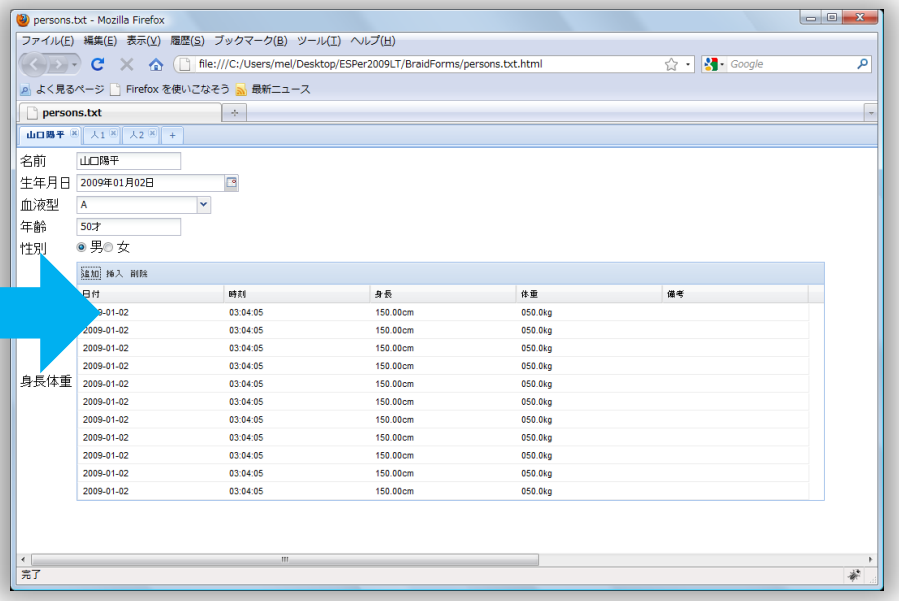
タスク	入力項目数	画面数
合計	4000 個	200 画面
人月毎	267 画面	13 画面

ユーザ言語で仕様を記述・整理する ＞ 効果

他のプラットフォームへの乗換

- **言語処理系を作りなすだけ**
  - 業務知識は全て再利用できる。
- **現在検討しているプラットフォーム**
  - .NET
  - AJAX
  - Android





ベンダーロック（良い意味で）

- **様々な理由**
  - お客様は自分で作った物に誇りと愛着を持つ。
  - お客様は試行錯誤になれるとやめられない。
  - DSLで効率化された両社関係は捨てるには惜しい。
  - DSLの言語処理系を作れる会社は多くない。
  - 別のシステムを作るとき非常に効率的で見通しが良い。
    - 生産記録スキーマ（を加工したもの）に情報を足して、新たな言語処理系を作ればよい。
    - 非常に有益な構造化がなされた多くの業務知識が再利用できる。
    - 具体的な情報をこねくり回してから判断できる。
    - 汎用言語プログラムか自然言語の仕様書の中に拡散する非定形な情報ではこうはいかない。

スーパーアジャイルじゃない？

- **抽出フェーズからPDCAサイクルが速い**
  - 私たちの動くものはプログラムだけじゃない。しっかりした意味を持つ言語を作るから、お客様の記述を私たちがインタプリートして身振り手振りで動かせる。いきなり現物確認できる。
- **最後はＰＤＣＡサイクルがめちゃめちゃ速い**
  - お客様だけですぐに画面（現物）が確認できる。