チャットボットシステムにおける
開発と運用の効率化に関する研究

提出先　　大阪大学大学院情報科学研究科

提出年月　2022 年 7 月

岩崎 信也

研究業績

A. 学術論文誌論文

1. 岩崎信也, 角田朋, 関口悦博, 小西幸洋, 大鳥朋哉, 薦田憲久, “アウトソース型セキュリティセンタにおけるインシデント対応迅速化のためのアラート調査支援システム,” 情報処理学会論文誌, vol. 60, no. 4, pp. 1108–1118, 2019.

2. 岩崎信也, 五十嵐智, 飯倉健, 津村直哉, 薦田憲久, 藤原融, “チャットボットと外部システムとの連携のためのシナリオ生成方式,” 電気学会論文誌 C, vol. 142, no. 8, pp. 919–928, 2022（掲載決定）.

B. 国際会議

1. S. Iwasaki, T. Kakuta, Y. Sekiguchi, Y. Konishi, T. Ohtori, and N. Komoda, “A Clustering - based Judgment Method of False Positive Alerts,” in *Proceedings of the 8th International Conference on Theory and Practice in Modern Computing 2019 (TPMC 2019)*, pp. 174–180, 2019.

C. 学会講演

1. 岩崎信也，角田朋，関口悦博，大鳥朋哉，薦田憲久，“アウトソース型セキュリティセンタにおけるインシデント対応迅速化のためのアラートログ可視化システム,”コンピュータセキュリティシンポジウム2017論文集，pp. 961–965, 2017.

2. 岩崎信也，角田朋，関口悦博，小西幸洋，大鳥朋哉，薦田憲久，“不正侵入検知におけるセキュリティアラートのシグネチャ別発生量に着目した誤検知削減手法,”コンピュータセキュリティシンポジウム 2018 論文集, pp. 441–442, 2018.

3. 岩崎信也，薦田憲久，藤原融，“チャットボットにおけるシナリオの管理容易化のための動的実行方式，” 第 19 回情報科学技術フォーラム（FIT2020）講演論文集，第 4 分冊, O-001, pp. 161–164, 2020.

4. 岩崎信也，薦田憲久，藤原融，“チャットボットにおけるポリシーによるシナリオチェック方式，” 情報処理学会第 84 回全国大会講演論文集，第 4 分冊, 2G-03, pp. 441–442, 2022.

D. その他

1. 岩崎信也，津村直哉，“身近になった対話システム：3. チャットボットサービスの変遷とそれを支える構成技術 ―シナリオ型チャットボットサービスの発展―,” 情報処理, vol. 62, no. 10, pp. e12–e18, 2021.

内容梗概

チャットボットは人との親和性が高く，多種多様な企業，公的機関において利用が進んでいる．チャットボットは，それで遂行するタスクに合わせて，利用者との対話の内容や対話の流れを定義したシナリオと呼ばれる対話テンプレートに基づき利用者と対話する．

チャットボットを新規に構築する場合，遂行したいタスクに合わせてチャットボットと利用者との対話内容を設計し，シナリオを作成する．しかし，最近は遂行するタスクに対する実行支援手段として Web システムや電話対応支援システムなどが既にあるところに，並行でチャットボットを導入するケースが現れてきており，作成後にタスクの変更があった際には，チャットボットのシナリオと既存の実行支援手段を実現しているシステムにおける対応手順を同時に更新する必要が生じる．また，業務担当者がシナリオをテスト，デバックできることが求められている．しかし，業務担当者はプログラミングに関する知識が十分でないことが多く，テスト方法の簡単化が不可欠である．

チャットボットを運用する場合，チャットボットはインターネット上に公開するシステムであり，サイバー攻撃の対象となる．このため，チャットボットの運用時，サイバー攻撃に備えるセキュリティ管理が必要となる．サイバー攻撃への対応として，運用しているチャットボットをファイアウォールや侵入検知システムなどのセキュリティ機器で監視し，発生するアラートを調査する．アラートの形式がセキュリティ機器により異なることや，調査のためのコマンドの作成，発生アラートの流れを把握するための整形などの問題により，迅速化の妨げとなっている．

チャットボットを効率的に開発できること，チャットボット運用時のセキュリティ管理を効率化させるため，本論文では，以下の方式の実現を目的とする．

課題(1)： チャットボット開発において，既存の実行支援手段を実現しているシステムに保存されている対応手順などの情報をシナリオ定義として簡単に利用できる方式

課題(2)： チャットボット開発の知識がないシナリオの作成者でも，シナリオのテストを容易に実施できる方式

課題(3)： セキュリティ管理におけるアラート調査の迅速化のための支援方式

本論文は全 5 章から構成される．

第 1 章の序論では，チャットボットの開発と運用フェーズを説明し，効率化に向けて解決するべき課題について述べ，従来研究を概観するとともに，本論文の目的と位置づけを明らかにする．

第 2 章では，既存手段を実現しているシステムのデータを利用して対話の流れを含めたシナリオを生成するフロー連携方式を提案する．既存手段を実現しているシステムのデータ構造は対象のシステムにより異なるため，シナリオの作成時に，既存手段を実現しているシステムのデータ構造をチャットボットのシナリオの構造に対応付けられるようにする．これにより，利用者の対話時に既存手段を実現しているシステムのデータから，シナリオを自動生成する．また，提案方式と従来方式でシナリオの作成時間を比較し，提案方式の効果を評価する．

第 3 章では，対話時の正しい動作を簡単にシナリオに記述できるようにすることで，シナリオが誤っている可能性がある箇所を機械的にチェックする方式を提案する．作成不良で起こる異常な動作に着目し，シナリオに作成不良がない場合の動作の条件を動作条件としてシナリオに紐づけ，シナリオの構造から機械的に動作条件に準拠しているか判定することで，シナリオを自動でテストする．これにより，シナリオ作成者によるテスト時間を減らし，作成不良の見逃しを減らす．実際に提案方式を利用することでシナリオの作成時に発生した作成不良をどの程度発見できるのかを実験し，提案方式の効果を評価する．

第 4 章では，チャットボットを運用する際のセキュリティ管理におけるアラート調査のための支援システムを提案する．アラート調査のための統一的なアラート形式を定義し，セキュリティ機器から発生するアラートの形式を統一的な形式に変換する．収集した統一的なアラートを 2 段階に構造化させることで局所的な発生と継続的な発生を可視化し，アラートの発生の流れの把握を分かりやすくする．また，アラート調査に必要な標準的な集計処理をアラート発生時に事前に実行することでコマンド入力の手間を削減する．実際のアラート調査時の必要時間を提案システムと既存システムを比較することで提案システムの効果を評価する．

第 5 章では，結論として本研究で得られた成果を要約し，今後の課題を述べる．

目次

第1章
序論

1.1　研究の背景

チャットボットは，対話型 UI（User Interface）を用いて，人とコンピュータが対話できるシステムであり，人と人との対話に近い能力を有することから，人との親和性が高い[1], [2]．チャットボットは，従来，雑談などの会話自体を目的としていたものが多かったが[3] [4]，2016 年以降，スマートフォンの普及とともに，チャットボットを用いて多種多様な企業, 公的機関において, マーケティング[5], 商取引[6], [7], 医療[8], [9], 教育[10], [11]，旅行[12], [13]，行政手続き[14], [15]などのサービスの利用が進んでいる．この結果，チャットボットへの関心が急速に高まっている[16], [17].

このため, 多種多様なタスクの遂行を目的として様々なユースケースに対応するためのチャットボットを開発・運用するチャットボットシステムとして，Google 社「Dialogflow[18]」，マイクロソフト社の「Bot Framework[19]」，LINE 社の「LINE[20]」，IBM 社の「Watson[21]」などが利用されている．チャットボットの利用シーンが日々多様化していくなかで, これまでのチャットボットシステムでは想定していなかった利用シーンが現れており, これらの利用シーンに向けたチャットボットの開発と運用の効率化に対するニーズが高まっている.

チャットボットは, シナリオと呼ばれる対話テンプレートに基づき利用者と対話する. シナリオは，ダイアログや対話テンプレート，会話フローとも呼ばれ，チャットボットで遂行するタスクに合わせて,「利用者に対してどのような質問を提示するか」「利用者の返答をどこに保存し，どのように処理し，次にどのような質問をするか」といった, 利用者との対話の内容や対話の流れを定義する[22], [23]. シナリオを作成するうえでのチャットボットで遂行するタスクの例を図 1-1 に示す. この例は, 利用者に名前と生年月日を聞き，年齢を計算後，20 歳未満の場合は申請不可とするタスクである.

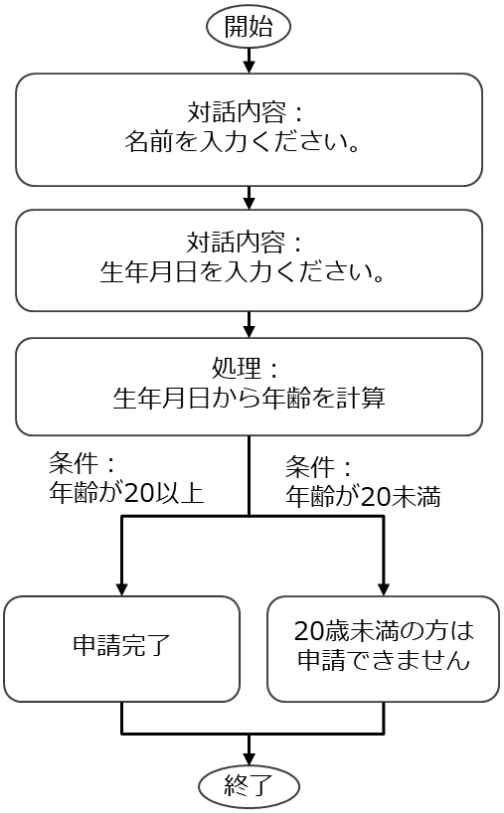

図 1-1 20 歳以上を対象とする申請処理タスクの例

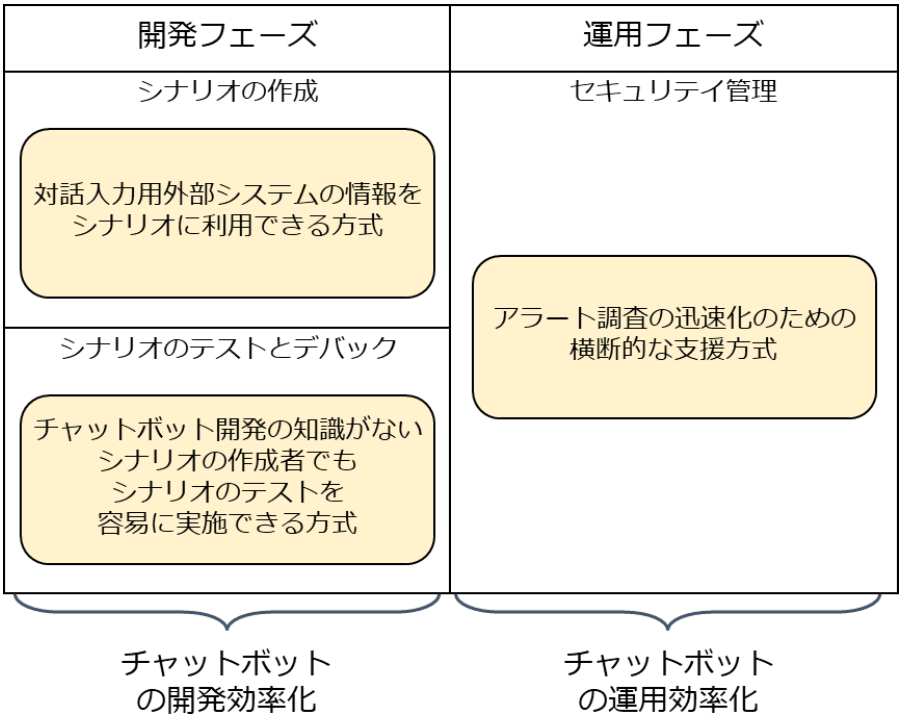

図 1-2 チャットボットの開発と運用フェーズにおける機能と課題

標準的なチャットボットシステムによるチャットボットの開発と運用を図 1-2 をもとに説明する．

(1)　シナリオの作成

チャットボットを提供する場合，まず，遂行したいタスクにあわせてチャットボットと利用者との対話内容を設計し，シナリオを作成する．チャットボットは他の WEB システムなどに比べ，小規模なシステムとして，手軽に短期間で提供できることが求められている．これに対応するため，シナリオの作成者は，専用のグラフィカルなインターフェース（GUI: Graphical User Interface）を用いるツール（以後，シナリオ作成ツールと呼ぶ）[24], [25]を利用してシナリオを作成することが多い．この結果，シナリオ作成の簡単化が進み，チャットボットの専門知識を持たない（チャットボットで遂行したいタスクの）業務担当者がシナリオの作成を担当することが増えている．

また，最近は遂行するタスクの既存の実行支援手段として，Web システムや電話対応支援システムなどがあるところに，利用者の利便性を目的として，並行でチャットボットを導入するケースが現れてきている．このようなケースでは，既存の実行支援手段を実現しているシステム（以後，このシステムを対話入力用外部システムと呼ぶ）と，チャットボットのシナリオで，それぞれ利用者への対応手順が定義される．これらの対応手順が独立に存在する場合に手順を変更しようとすれば，チャットボットのシナリオと対話入力用外部システムの対応手順の両方を変更する（同期をとる）ことが必要となる．それらの定義を人手で同期しようとすると，手間が増加する．また，対話入力用外部システムとチャットボットで管理者が異なることが多く，対話入力用外部システムの対応手順の変更を適切にシナリオに反映できない，同期が遅れるといったミスの発生につながる．

(2)　シナリオのテスト・デバッグ

作成したシナリオは，シナリオの作成者の意図通りに動くかをテストし，何らかの問題があればそれをデバッグする必要がある[26], [27]. テストで一般的によく用いられる方法としては，シナリオの作成者が，作成したシナリオを元に実際に対話型 UI を利用してチャットボットと利用者の対話を模擬することで，その対話結果から仕様通り動作するか確認する方法がある．対話結果から仕様外の異常な動作が見つかった場合，その動作の原因となるシナリオ上の作成の誤り（以後，作成不良と呼ぶ）を特定し，シナリ

オを改善する．これを異常な動作が見つからなくなるまで繰り返す．

最近では，業務担当者がシナリオを作成することが増えてきた結果，チャットボットの開発効率化のため，シナリオの作成と同じく業務担当者がシナリオをテスト，デバックできることが求められている．しかし，業務担当者がシナリオをテスト，デバックするには，テスト方法の簡単化が不十分であり，テストを完全に実施できないという問題が起こる．特に，対話の中にループや条件分岐が多い場合は，シナリオの流れや発話内容の正しさを確認するのに，何度も模擬対話を繰り返すことが多く，迅速化の妨げとなっている．さらにこのような状況では模擬対話を十分に行うことができない場合もあり作成不良の見逃しが起こりやすい．

(3)　セキュリティ管理

チャットボットは，インターネットに公開するシステムであり，サイバー攻撃の対象となる．このため，チャットボットの運用者は，チャットボットを運用する際に，サイバー攻撃に備えて，セキュリティを管理する必要がある[28]．具体的には，セキュリティ情報を収集し，脆弱性が発見された場合にはシステムへの影響度を確認したうえで，必要に応じてシステムを改善する[29], [30]．さらに，チャットボットを監視し，サイバー攻撃の可能性を検出，セキュリティインシデント（以下，単に「インシデント」と記す）とならないように対応するインシデント対応が重要となっている[31]–[33]．インシデントは,「情報システムの運用におけるセキュリティ上の問題として捉えられる事象」を指す[34]．例として，サイバー攻撃による情報流出や不正アクセス・ウェブサイト改ざんなどがある．これらインシデントが発生し，対応が適切でないと，サービス提供元のブランドに悪影響を与え，多額の賠償金が発生する場合もある．

インシデント対応では，インシデントの疑いが検知された際に，優先度の決定や対応が必要かを判断するトリアージが重要となる．トリアージでは，ファイアウォールや侵入検知システムといったセキュリティ機器が発生させるアラートを調査する．これらのセキュリティ機器は，ネットワークトラフィックを監視し，異常や攻撃の可能性が高いイベントを検知した際にアラートを発生させる．このアラートは多量かつその大部分が問題ないことが分かっている[35], [36]．しかし，アラート調査では，アラートの形式がセキュリティ機器により異なることや，調査のためのコマンドの作成，発生アラートの流れを把握するための整形などを問題として，時間を要し迅速化の妨

げとなっている．このためアラート調査を迅速化しセキュリティ管理を効率化させることが求められている．

このように，チャットボットの開発時にチャットボットを効率的に開発できること，チャットボットの運用時にセキュリティ管理を効率化させることが望まれている．これに対処するため，本論文では以下のとおり，チャットボットの開発時の課題(1)および課題(2)，チャットボット運用時の課題 (3)の方式の実現を目的とする．

課題(1)： シナリオの作成において，対話入力用外部システムに保存されている対応手順などの情報をシナリオ定義として簡単に利用できる方式

課題(2)： チャットボット開発の知識がないシナリオの作成者でも，シナリオのテストを容易に実施できる方式

課題(3)： セキュリティ管理におけるアラート調査の迅速化のための支援方式

1.2 関連研究

1.2.1 対話入力用外部システムの情報をシナリオに利用する方式

対話入力用外部システムを含むチャットボット外のシステムからデータを取得し対話に利用できるチャットボットとして多くの研究が進んでいる．文献[37]–[39]では，対話入力用外部システムに格納されているデータを，API（Application Programming Interface）を利用して取得しシナリオ内の一部の要素に利用する方式を紹介している．シナリオの作成時にシナリオ内の利用者に対する発話文などに発話するテキストの代わりに，対話入力用外部システムのデータを参照する．これにより，利用者の対話時に，対話入力用外部システムから参照されたデータを取得することでチャットボットの発話などに利用する．しかし，シナリオ専用のデータ構造と対話入力用外部システムのデータ構造が異なるため，シナリオのすべての要素に対話入力用外部システムのデータを利用することは出来ず，この方式では対話入力用外部システムのデータを利用できるのはシナリオ内の一部の要素にとどまる．

質問対応のユースケースへ限定したチャットボットにおいて，自動的にチャットボットのシナリオを生成する方式を提案している事例もある[40]–[42]．質問対応のチャットボットとは，1 問 1 答形式で利用者の質問に対して，チャットボットが事前に準備され

た回答群から最も適した回答を返すものであり，継続した対話はできない．この事例では，既存の「ヘルプページ」や「よくある質問」といったWEBページのデータを読み込むことで対話入力用外部システムの情報から，専用の機械学習モデルによりシナリオを生成する．この手法は，質問対応のユースケースでのみ利用可能な手法であり，それ以外の情報収集などの様々なタスクの遂行を目的としたチャットボットでは利用が難しい．

チャットボットと外部システムを同時導入することを前提とした連携事例もある．旅行代理店のチャットボットとして，旅行システムと連携することで利用者の希望にあった旅行プランを推薦するシステム[43]，利用者の希望に応じた医薬情報を医薬情報システムから提示するシステム[44]，社内情報サービス管理システムと連携し，サービスの利用開始や利用停止などを簡単に実施するシステム[45]，システム開発における負荷テスト時に，性能負荷テストツールと連携してテストの設定や結果の解釈を簡単にするシステム[46]などがある．これらの研究は，チャットボットとの情報共有を意図した専用の外部システムとチャットボットを同時に導入する場合を想定しており，既存の手段として対話入力用外部システムがあるところに追加でチャットボットを並行で導入することを想定していない．

また，対話入力用外部システムのデータをチャットボットのシナリオに利用するには異なる形式データの変換が必要となり，変換にはETL（Extract/Transform/Load）ツールやローコードツールの利用事例がある[47]–[50]．これらのツールは，データ変換処理の開発者が，変換元，変換先双方のデータ構造を理解して使用する必要がある．しかし，チャットボットの開発においては，シナリオの作成者は，シナリオ作成ツールによりグラフィカルにシナリオを作成するため，シナリオのデータ構造を知ることができず，これらのツールの利用が難しい．また，これらのツールは，効率的に変換処理を作成するためのパーツが提供されており，それらを利用して変換元から変換先にどのように変換するかの処理を作り込む必要があり，ツールの知識に加えてプログラミングの知識がない場合は利用が難しい．

1.2.2 シナリオのテストを容易に実施する方式

シナリオのテストの効率化方法として，文献[51]では，チャットボットの簡易的なエミュレータを用いてシナリオ作成者によるテストを効率化する方法を紹介している．このエミュレータは，作成中のシナリオを利用して対話を模擬することができ，複数の利用者を模擬した同時対話や対話内容を記録する機能を持つ．シナリオ作成後の模擬対話時に必要なチャットボットへのシナリオ適用や，模擬対話の準備を効率化することができるが，模擬対話自体はシナリオの作成者が，従来通り実施する必要があり，テストにかかる時間の削減や作成不良の見逃しの対応としては不十分である．

また，システム開発の一般的なテスト方法の一つとして，入力値とそれに対する出力の期待値を指定したテストケースを事前に与えておくことでシステムの開発・改修時に自動でテストする方法がある[52]–[54]．文献[55]–[57]では，シナリオの作成者が，利用者による一連の想定発話文と，想定発話文に対する期待されるチャットボットの返答結果のリストを事前に指定することで，自動でシナリオをテストするシステムを提案している．これらの方法は，一般的なシステム開発でのテスト方法をチャットボットに適用しているものであるが，前述したとおり最近のシナリオ作成者は，業務担当者でありチャットボットやソフトウェア開発の知識を持たないことが多く，テストに必要なすべての想定発話文と期待される返答結果のリストを厳密に指定するのは難しい．また，シナリオの更新の際にはそれらのリストも正確に更新する必要があり管理の手間がかかる．

1.2.3 アラート調査の迅速化のための支援

アラート調査の迅速化として，これまでチャットボットなどの監視対象のシステムに関連したトラフィックや発生したアラートを特定の観点で可視化する研究が盛んである．通信元および通信先の傾向を二次元マップにより可視化する事例[58]，監視センサーとダークネットとの通信を可視化しマルウェアの伝搬を可視化する事例[59]，アラートをグループ化しその相関を視覚的に可視化する事例[60]，複数の IDS のアラートを関連付けて可視化する事例[61]などがある．これらの研究は，特定の攻撃や異常を発見するために，特定の観点でアラートやトラフィックを可視化する手法を提案している．しかし，実際のインシデント対応では，さまざまな攻撃の可能性を分析する上で，分析担当者による多角的な観点での迅速な調査が必要である．このため，特定の観点で可視

化するこれらの研究では，アラート調査に必要な観点を網羅できないという課題がある．

複数のログを可視化し，時系列的な出現頻度を視覚化する事例[62], [63]もある．これらの研究では，クライアントデバイスのログを含む多種のログを可視化することで，同一の時間帯に発生したログやその頻度を横断的に調査することが出来る．しかし，クライアントデバイスのログは，そのサイズの点で，コストが高いため，収集していないことが多く，アラートが発生するたびにクライアントデバイスを含む多種のログを収集して調査することは難しい．また，機械学習などにより機械的に誤検知を削減する研究が盛んである[64]–[68]．例えば，アラートを含めたさまざまな情報を集約して異常検知する事例[69]，アラートを比率分析・しきい値学習分析などの統計手法により誤検知を削減する事例[70]もある．これらは，誤検知の可能性が高いアラートを提示することを目的としている．しかし，近年のチャットボットなどの公開システムに対するサイバー攻撃は，多様かつ巧妙化しており機械的な分析のみではインシデントの判断が難しい．

アラートの形式が異なる問題への対応としては，syslog で送信されたログを，正規表現を用いて指定の形式に変換，格納する事例[71]がある．これは，提案されている可視化手法のためのアラート収集方式である．そのため，転送方式は syslog であり，収集対象となる項目が決まっている．実運用されているセキュリティ機器は，設置されているネットワークにより，必ずしも syslog を利用できるわけではない．また収集対象項目が，固定的かつ提案する可視化手法に必要な最低限の項目であり，アラート調査観点を調べるために必要となる項目を充足していない．

アラートのフォーマットとして，STIX (Structured Threat Information eXpression)[72]や，IDMEF (Intrusion Detection Message Exchange Format)[73]がある．双方とも多種のシステム間でのデータ交換を目的としたフォーマットであり XML で構成され，XML タグを定義する必要がある．そのため，アラート 1 件当りのサイズが大きくなり，アラートの発生量が 1 日で数万件になるアラート調査用のアラートデータ蓄積には適さない．また，汎用的なログ収集ソフトウェアとしては，fluentd[74]や logstash[75]がある．これらのソフトウェアは，多種の機器からログを収集し，変換し，他のシステムに，転送することができる．しかし，これらのソフトウェアでアラートを収集する場合，複雑な定義ファイルを必要とし，これらのソフトウェアの知識を持たない場合，利用が難しい．また，アラートの変換では，アラートと他のファイルを対応づけて，項目を追加する処理が必要となるが，既存のソフトウェアでは対応できない．

1.3　研究の方針

前節までに述べた課題を踏まえ，本研究における各課題の位置づけとアプローチを図 1-3 に示す．

(1) チャットボットと対話入力用外部システムとの連携のためのシナリオ生成方式

課題(1) に対して，対話入力用外部システムのデータを利用して対話の流れを含めたシナリオ全体を生成できる方式を提案する．対話入力用外部システムのデータ構造は対象のシステムにより異なるため，シナリオの作成時に対話入力用外部システムのデータ構造をチャットボットのシナリオの構造に，表形式の定義で簡単に対応付けられるようにする．また，チャットボットでは既存手段ではなかった独自の対応手順が必要となることもあるため，対話入力用外部システムからの生成シナリオとチャットボット独自のシナリオを組み合わせられるようにする．具体的には，対話入力用外部システムのデータ構造とシナリオの各項目を対応付けるための表形式の変換定義と，利用者の対話時に指定された変換定義をもとに対話入力用外部システムのデータを取得しシナリオを自動生成するシナリオ生成方法を提案する．

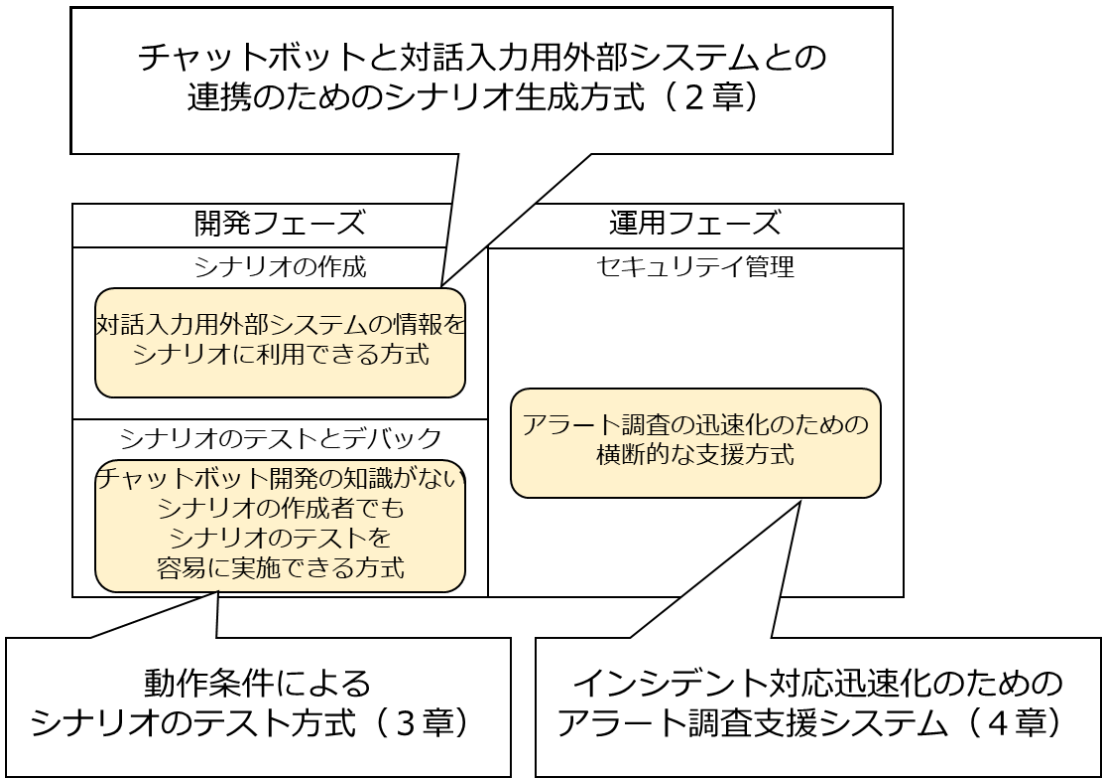

図 1-3　本研究における課題へのアプローチ

(2) 動作条件によるシナリオのテスト方式

課題(2) に対して，シナリオの作成者がシナリオのテストを容易に実施できるようにするため，対話時の仕様の動作を簡単にシナリオに記述できるようにすることで，対話の流れが誤っている可能性がある箇所を機械的に確認し，シナリオ作成者に提示する方式を提案する．これにより，シナリオ作成者による模擬対話によるテストの前に，できるかぎりシナリオの異常な動作を発見し，作成不良を解消することで，模擬対話にかかる時間を減らし，作成不良の見逃しを減らす．具体的には，シナリオのどこがどのような条件のときにシナリオとして正しいかを指定するシナリオ向きの動作条件定義方法と，動作条件違反の確認方法を提案する．動作条件定義方法では，シナリオの構成要素に対してシナリオに作成不良がない場合の動作の条件を動作条件として記述できるようにし，動作条件違反の確認では，シナリオの構造からシナリオが指定された動作条件に準拠しているかを確認する．

(3) インシデント対応迅速化のためのアラート調査支援システム

課題(3)に対して，アラート調査の各種問題を解決するための支援システムを提案し，アラート調査を迅速化する．このシステムでは，アラート調査のための統一的なアラート形式を定義し，各種セキュリティ機器から発生するアラートの形式を統一的な形式に対応付けることで変換する．収集した統一的なアラートをアラートの攻撃元の情報とアラート間の発生時間の間隔により 2 段階に構造化させることで局所的な発生と継続的な発生を可視化し，アラートの発生の流れの把握を分かりやすくする．また，アラート調査に必要な標準的な集計処理をアラート発生時に事前に実行することでコマンド入力の手間を削減する．必要があればインシデント対応の担当者が簡単なフォームで集計処理を実行できるようにする．

1.4 本論文の構成

本論文では，第 2 章以降を以下のとおり構成する．

第 2 章では，文献[76], [77]に基づき，対話入力用外部システムのデータを利用して対話の流れを含めたシナリオを生成するフロー連携方式を提案する．はじめに，チャットボットが連携する必要がある対話入力用外部システムの仕様について説明する．次に，

この対話入力用外部システムからチャットボットのシナリオを生成するための生成方式を説明する. 最後に対話入力用外部システムと連携するチャットボットのシナリオの開発工数を提案方式と従来方式を比較することで提案方式の効果を評価する.

第 3 章では，文献[78]に基づき，対話時の仕様の動作を簡単にシナリオに記述できるようにすることで，シナリオが誤っている可能性がある箇所を機械的にチェックし，シナリオの作成者に提示する方式を提案する．はじめに，チャットボットのシナリオで発生する作成不良とテスト手順について説明する．次に，シナリオの作成不良を機械的に発見するための確認方式として動作条件の定義方法や動作条件によるテストの実行について説明する．最後に，実際に提案方式を利用することでチャットボットのシナリオの作成時に発生した作成不良をどの程度発見できるのかを実験し, 提案方式の効果を評価する.

第 4 章では，文献[79]–[82]に基づき，アラート調査のための支援システムを提案する．はじめに，セキュリティ管理におけるサイバー攻撃対応の手順とその課題を説明する．次に，課題に対する提案システムのアプローチを説明する．最後に実際のサイバー攻撃対応時の必要時間を提案システムと既存システムを比較することで提案システムの効果を評価する.

第 5 章では，結論として本研究で得られた成果を要約し，今後の課題を述べる.

第2章
チャットボットと対話入力用外部システムとの連携のためのシナリオ生成方式

2.1　緒言

本章では，効率的なシナリオの作成のために，対話入力用外部システムのデータを利用して対話の流れを含めたシナリオを生成する方式を提案する．

商用でのチャットボットの適用では，導入企業側に Web システムや電話対応支援システムなどの既存のタスクの実行支援手段があるところに，並行で導入されることが多い．このため，チャットボットを既存の手段と独立して作ると，チャットボット用のシナリオ定義を重複して行う必要があり作成工数の増加を引き起こす．また，情報の更新時に両方の手段の情報を管理・更新しなければならず開発・運用工数の増加や，ミスの発生につながる．

これに対して，先行研究として，対話入力用外部システムからデータを取得し対話の一部に利用できるチャットボットの事例がある[37]–[39]．しかし，このチャットボットでは，利用できる情報が利用者への発話文など一部の情報に限られ，対話の流れなどシナリオの構造を対話入力用外部システムの情報によって変更することができない．

これを解決するため，本章では，対話入力用外部システムのデータを利用して対話の流れを含めたシナリオを生成するフロー連携方式を提案する．提案方式は，取得データの構造と対話の流れなどのシナリオの構造を簡単に対応づけるための表形式のシナリオ生成定義方法と，それにより定義される対話入力用外部システムのデータからシナリオを自動生成するシナリオ生成方法から構成される．これにより，対話入力用外部システムのデータを対話の流れなどのシナリオの構造にも利用できるようにする．

以下，第 2.2 節で，チャットボットのシナリオの説明と対話入力用外部システムと連携する上の問題点を説明し，第 2.3 節で，提案するフロー連携方式を説明し，第 2.4 節で，実際に対話入力用外部システムと連携するチャットボットのシナリオ作成を例に，提案方式の評価を行う．

2.2　チャットボットと対話入力用外部システムとの連携によるシナリオ作成

2.2.1 シナリオ

チャットボットでは，事前にシナリオと呼ばれる対話テンプレートを定義する必要がある．シナリオでは，チャットボットで遂行する「タスク」に合わせて，「利用者に対してどのような質問を提示するか」「利用者の返答によって，次の質問をどのようにするか」といった，利用者との対話の内容や対話の流れを定義する．なお，「タスク」とは，注文処理や申請処理などの，これまで電話対応や Web システムで実施していた一定の業務を指す．なお，本論文のチャットボットの用語は．基本的に日本マイクロソフト（株）のチャットボットサービスの用語に準拠している．

シナリオは，定義 2-1 に示すように複数の「ステップ」とステップからステップへの「遷移」から成るとする．定義 2-1 は，BNF 記法[83]を拡張した記法で示しており，「＋」は要素の組みを意味し，「|」は区切られた要素の選択を意味し，「＋」を後置している要素は 1 回以上繰り返す要素を意味する．

定義 2-1. シナリオの構成

シナリオ ::= (ステップ ＋ 遷移) ＋

ステップ ::= ID ＋ 発話文 | 対話 | 処理ロジック

対話　　::= 発話文 ＋ 選択肢

遷移　　::= (遷移条件 ＋ 遷移先 ID) ＋

「ステップ」・「遷移」について，それぞれ説明する．

(1)　ステップ

ステップは，チャットボットにおける一定の対話単位であり，ID が付けられる．「発話文」ステップ，「対話」ステップ，「処理ロジック」ステップの 3 種類から成る．

「発話文」は，チャットボットが一方的に利用者に発話する文章を定義したものであ

る．発話文には，スロットと呼ばれる利用者の選択結果や計算結果などの結果データを対話期間中保管する変数を含むことができる．スロットは，「slot["参照するステップのID"]」で記述し，発話文や遷移条件等で使用することができる．

「対話」は，「発話文」，「選択肢」のパラメータから成り，利用者に提示する「発話文」と「選択肢」を定義したものである．

「処理ロジック」は，スロットを含む計算処理を定義したもので，JavaScript 記法[84]で記述したものである．

(2) 遷移

遷移は「遷移元 ID」，「遷移条件」，「遷移先 ID」から成る．「遷移先 ID」は「遷移条件」とセットになって，「遷移条件」が成立した際の遷移先ステップを定義したものである．最低 1 組は定義する必要がある．「遷移条件」は，JavaScript 記法の条件式で記述する．条件式の記述には，スロットを使うことができる．

シナリオの例として 20 歳以上の人を対象とする申請処理シナリオを図 2-1 に示す．このシナリオでは，チャットボットの利用者に名前と生年月日を質問し，利用者への発話に対して返答された生年月日から年齢を計算し，20 歳以下であれば申請できない旨を発話している．このようなシナリオを作成する場合には，図 2-2 で示すようなシナリオ作成ツールが用いられることが多い．シナリオ作成ツールでは，一般的にシナリオの流れをグラフィカルにフローで表現し，「ステップ」や「遷移」の作成を容易にすることで，人手によるシナリオの作成を支援する．

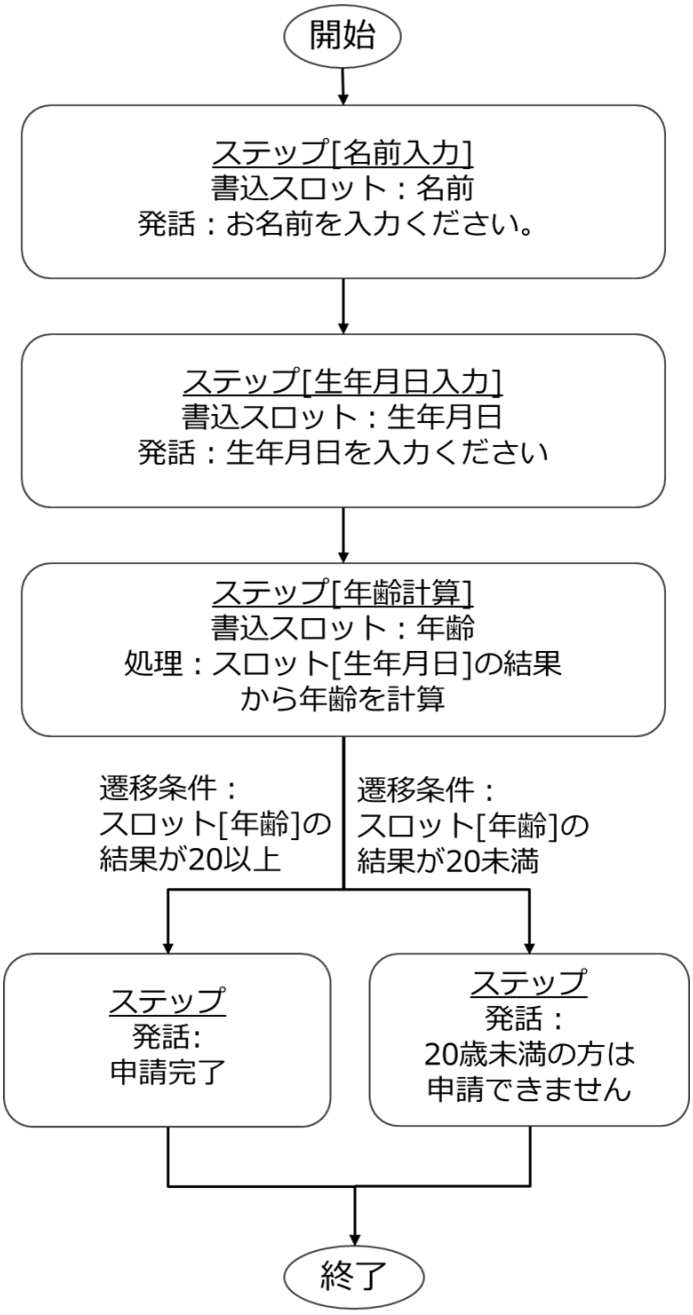

図 2-1 シナリオの例

（20 歳以上を対象とする申請処理シナリオ）

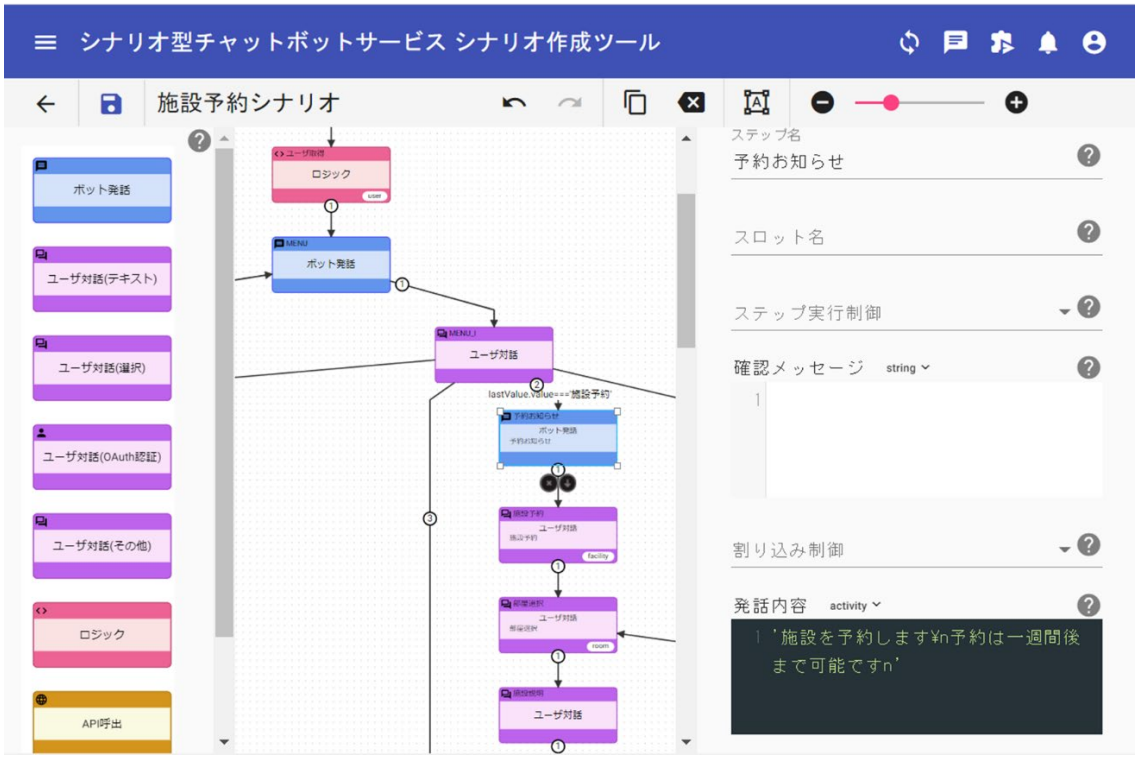

図 2-2 シナリオ作成ツールの例

2.2.2 チャットボットと連携する対話入力用外部システム

最近はタスクの既存の実行支援手段として，Web システムや電話対応支援システムなどが存在するときに，チャットボットを並行導入する場合が増えている．それらの既存の実行支援手段を実現している対話入力用外部システムは，近年企業や教育機関での利活用が進んでいるビジネスアプリ作成サービスやローコードアプリケーションプラットフォームと呼ばれるサービス[85]–[87]により作成されることが多い．ビジネスアプリ作成サービスは，データ管理やデータ入力・表示などの機能を持つシステムを GUI ベースで容易に作成できるサービスである．

これらの対話入力用外部システムからデータを取得したときの標準的なデータ構造には，テーブル形式と，オブジェクト形式がある．テーブル形式のデータ構造は，表データを列・行の 2 次元配列で表現したデータであり，列は同一の種類の情報，行は各列の情報の組み合わせが格納されている．1 行目はヘッダー行として，各列の情報の意味を表す項目名となっている場合もある．テーブル形式のデータの例を図 2-3(a)に示す．このデータは， 4 行*4 列の二次元配列であり，第 1 行はヘッダー行であり残りの 3 行

が実データ行である．

一方，オブジェクト形式のデータ構造は，関連のある情報が key-value 方式の連想配列で表現され，それらの集合が配列に格納される．テーブル形式と異なり，複雑な構造を持つ（1 要素の中にさらに複数の要素を持つ入れ子の）データも表現できる．オブジェクト形式のデータの例を図 2-3 (b)に示す．このデータは，3 つの要素を持つデータであり，各要素の「メニュー選択」の要素内にさらに複数の要素を持つ．

```
[
 [“メニュー名"                ,"選択1"  ,"選択2"    ,"選択3"],
 ["ピザを選択ください"        ,"チーズ" ,"ミックス"          ],
 ["トッピングを選択ください","チーズ" ,"サラミ"            ],
 ["飲み物を選択ください"      ,"コーラ" ,"ビール“   ,"お茶"  ],
]
```

(a) テーブル形式.

```
[
 {
  メニュー名:ピザ選択,
  メニュー文:ピザを選択ください,
  メニュー選択：[{表示: チーズ　,次のメニュー:飲み物選択},
                 {表示: ミックス,次のメニュー:トッピング選択}]
 },{
  メニュー名:トッピング選択,
  メニュー文:トッピングを選択ください,
  メニュー選択：[{表示: チーズ,次のメニュー:飲み物選択},
                 {表示: サラミ,次のメニュー:飲み物選択}]
 },{
  メニュー名:飲み物選択,
  メニュー文:飲み物を選択ください,
  メニュー選択：[{表示: コーラ},
                 {表示: ビール},
                 {表示: お茶}]
 }
]
```

(b) オブジェクト形式.

図 2-3 外部システムのデータ構造

2.2.3 対話入力用外部システムとの連携によるシナリオ作成の問題点

対話入力用外部システムが既存で存在し，追加でチャットボットを導入するような状況では．対話入力用外部システムとチャットボットでそれぞれ利用者への対応方法が定義される．それらの定義を人手で同期を取ると，シナリオの作成・修正工数が増加する．また，対話入力用外部システムとチャットボットで管理者が異なるため，対話入力用外部システムの変更を適切にシナリオに反映できない，同期が遅れるといったミスの発生につながる．

そこで，既存の実行支援手段を実現している対話入力用外部システムと連携できるチャットボットが文献[37]–[39]で紹介されている．この技術を活用すると，対話入力用外部システムに格納されているデータを API 経由で取得し，シナリオ内のステップの発話文や選択肢などのパラメータに利用することができる．シナリオの作成時にステップの発話文等に発話する文を指定する代わりに，対話入力用外部システムのデータを参照する．これにより，対話入力用外部システムから参照されたデータを取得し利用者との対話に利用する．ステップ内のパラメータを対話入力用外部システムと連携できる特徴があることから，本研究で提案するフロー連携方式と区別するため，この方式をパラメータ連携方式と呼ぶ．

パラメータ連携方式での対話入力用外部システムからシナリオへのデータ取得のイメージを図 2-4 に示す．図 2-4 左は料理注文のチャットボットシナリオである．各店舗（ピザ店，弁当店）のメニュー情報が対話入力用外部システムにテーブル形式で保存されている．ここではピザ店への注文を例に説明する．ピザ店への料理注文は，「店舗選択」・「ピザ選択」・「トッピング選択」・「飲み物選択」・「発注処理」の 5 ステップから構成される．「ピザ選択」・「トッピング選択」・「飲み物選択」ステップの発話文・選択肢は，対話入力用外部システムのデータを参照しており，対話入力用外部システムからデータを取得し，利用者に発話される．対話入力用外部システムのデータが更新された場合，チャットボット上の発話文も変更される．

しかし，パラメータ連携方式では，シナリオ専用のデータ構造と対話入力用外部システムのデータ構造が異なるため，対話入力用外部システムのデータを利用できるのはステップ内部の発話文・選択肢といった一部のパラメータにとどまり，対話入力用外部シ

ステムのデータに，ステップ ID・発話文・選択肢といった「ステップ」を生成できる情報や，遷移条件・遷移先 ID といった「遷移」を作成できる情報が含まれるケースでも，チャットボット専用のシナリオを個別に作成する必要がある．図 2-4 左のシナリオで，ピザ店以外の弁当店のシナリオを追加する場合や，ピザ店の注文フローを変えようとした場合など，シナリオ作成者が事前に手動で各店舗のシナリオをそれぞれ作成し，メンテナンスする必要がある．このため，シナリオの作成工数やメンテナンス工数の増加につながる．また，シナリオをメンテナンスするのに，シナリオの知識を持つ技術者が必要となってしまう．

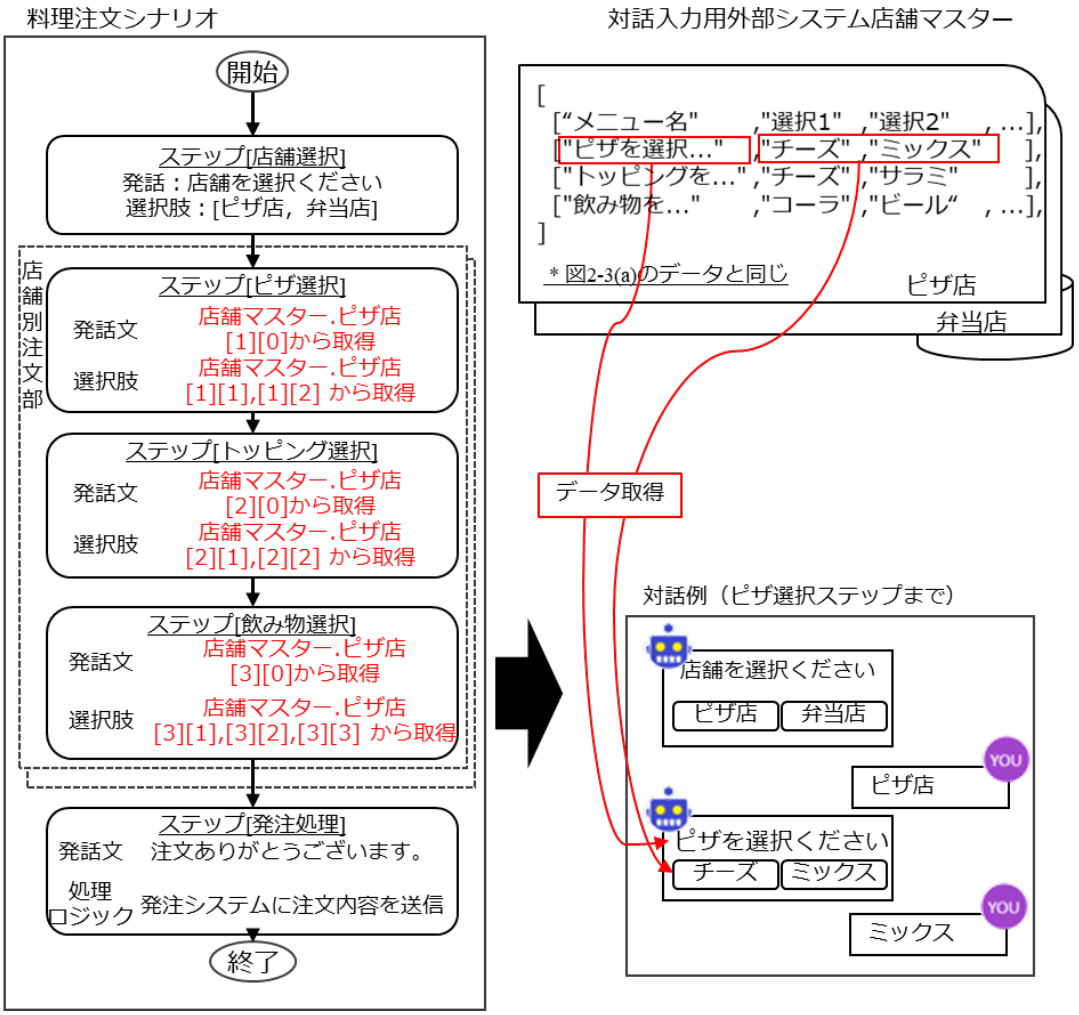

図 2-4 パラメータ連携方式による外部データの取得

2.3　対話入力用外部システムのデータからシナリオを生成するフロー連携方式

2.3.1　フロー連携方式の概要

本章では，対話入力用外部システムにシナリオの「ステップ」や「遷移」を生成できる情報があり，その情報を API により取得できることを前提に，シナリオの流れを定義せずに，それらの情報からシナリオを生成する方式を提案する．以後，提案方式をフロー連携方式と呼ぶ．

フロー連携方式では，「ステップ」の種類に，フロー連携用の専用ステップ（以後，フロー連携用ステップと呼ぶ）を新たに増やす．フロー連携用ステップは，「ID」と「変換定義」から成る．変換定義の詳細は 2.3.2 節で説明するが，これにより対話入力用外部システムのデータ構造とシナリオの構造を表形式の定義で簡単に対応付けられるようにする．変換定義は，シナリオの作成者がシナリオ作成ツールにより，シナリオ内にフロー連携用ステップを作ることで，その構成情報として記述できる．変換定義を記述する際は，変換定義の指定ミスや設定時間を削減するため，変換定義の入力候補を表示することで，シナリオ作成者による変換定義の指定を容易にする．具体的には，シナリオ作成ツール上で対話入力用外部システムのデータを実際に取得し，変換定義内容を自動設定したり，指定候補をプルダウン形式で表示したりすることで，シナリオの作成者が確認・選択できるようにする．

図 2-5 に示すように，チャットボットと利用者の対話時に，シナリオ内のフロー連携用ステップの変換定義に基づき，対話入力用外部システムのデータから自動でシナリオを生成する．変換定義の指定ミスなどでシナリオの生成に失敗した場合，シナリオの作成者に通知される．生成されたシナリオは，対話中のシナリオ内のフロー連携用ステップと置き換えられ，対話中のシナリオと組み合わさり，対話システムでの利用者との対話に利用される．フロー連携用ステップをシナリオの 1 ステップとすることで，対話入力用外部システムにデータがないチャットボット特有のシナリオと，対話入力用外部システムから生成したシナリオを組み合わせることができるようにする．利用者とチャットボットの対話時にフロー連携用ステップが実行されることで，シナリオが生成されるため，実行時点の対話入力用外部システムのデータからシナリオを生成できる．

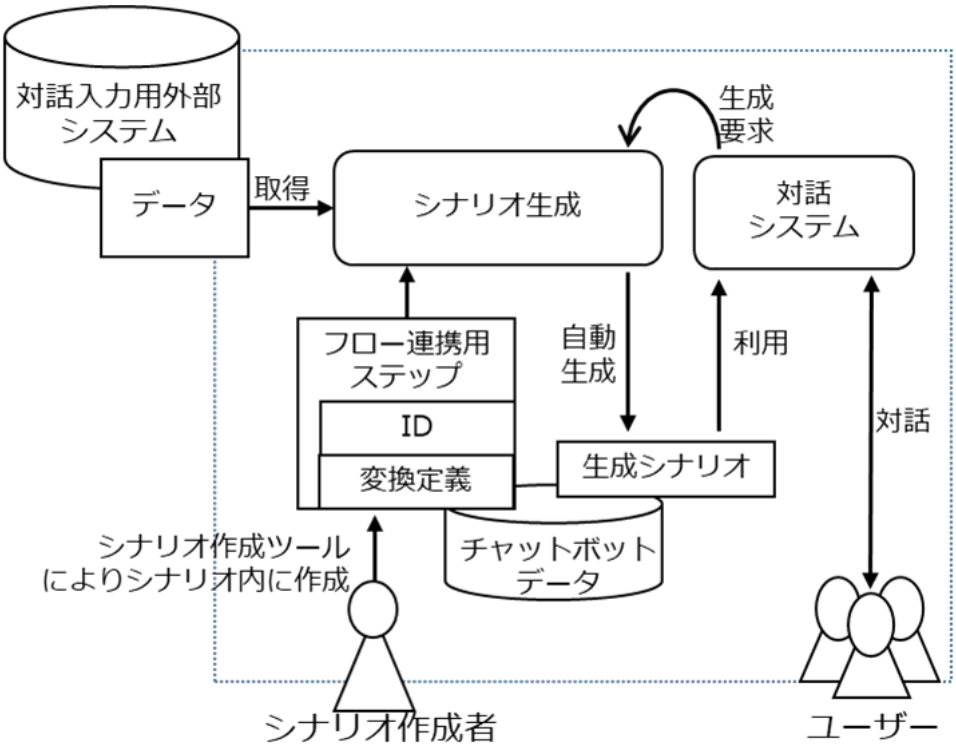

図 2-5 提案方式のモデル

シナリオ生成では，利用者による入力を修正するシナリオの作成を支援するために，生成シナリオを自動で拡張できるようにする．電話や Web システム等の対話処理では，利用者の入力内容を確認・修正した後，次の質問等の処理に移ることがよく行われる．同等のことをチャットボットで実現する場合は，利用者に対話結果（スロットに格納されている利用者の発話内容）を確認するステップを作成し，その後修正希望があれば，再び入力のためのステップを実行する．料理の注文シナリオの例では，最後に注文内容を利用者に提示し，間違っていれば前に実行したステップに遷移することで注文内容を再入力する．このとき，利用者の不要な再入力を避けるため，既回答を表示し修正するかどうか利用者に聞き，修正が必要な場合のみ，再入力できるようにすることが多い．フロー連携方式でこの方法を実現するには，対話入力用外部システムのデータを編集する必要があり既存の実行支援手段に影響を与える可能性があるため難しい．そこで，生成シナリオでステップごとに既回答を表示し修正が必要な場合は再入力できるようにシナリオを自動的に拡張することで，容易に対話結果を修正できるようにする．

変換定義については 2.3.2 節で，定義されたものをシナリオに変換する方式を 2.3.3 節で，変換定義の作成支援について 2.3.4 節，利用者入力の確認修正ステップの自動生成について 2.3.5 節で，それぞれ詳しく説明する．

2.3.2 変換定義

変換定義は，対話入力用外部システムのデータに，ステップ ID・発話文・選択肢といった「ステップ」を生成できる情報，遷移条件・遷移先 ID といった「遷移」を作成できる情報があるケースで，対話入力用外部システムのデータとシナリオの構造を対応付ける．対話入力用外部システムのデータ構造がテーブル形式の場合は表 2-1 に示す項目を，オブジェクト形式の場合は表 2-2 に示す項目を指定する．

最初に，テーブル形式に対応する表 2-1 を説明する．「取得先 API」には，対話入力用外部システムからデータを取得するための API の URL を指定する．取得先 API には，スロットを使うことができる．「データ構造」は，「テーブル」を指定する．「マッピング情報」はシナリオと対話入力用外部システムのテーブル形式のデータを対応付けるための情報である．取得データの何列目に，発話文や選択肢のデータがあるのかを対応付けるため，「発話文列番号」「選択肢 1~n 列番号」に取得したテーブル形式のデータの，シナリオの発話文・選択肢を意味するデータがある列番号（0 ~n 番目）を指定する．対話入力用外部システムのデータによっては，テーブル形式のデータの 1 行目がデータの項目名であり，ステップの生成対象のデータとはならないため，「ヘッダー行」に「有」を指定することで，取得データの 1 行目を生成対象から外すことができる．また，生成されるステップの ID は指定せず，取得データの行番号を ID とする．

次に，対話入力用外部システムのデータがオブジェクト形式の場合，データの項目名や構造が対話入力用外部システムにより異なるため表 2-2 に示すオブジェクト形式向けの変換定義を使用し，どこにシナリオの生成に必要となるデータがあるかを対応付ける．「API」には，テーブル形式と同じく対話入力用外部システムからデータを取得するための API の URL を，「データ構造」は「オブジェクト」を指定する．ステップの生成に必要となる「ステップ ID」，「発話文」，「選択肢」，「選択肢文」，「遷移先 ID」に対応する対話入力用外部システムのデータ構造の識別子を指定する．「ステップ ID」に生成するステップに付ける ID の識別子を指定する．「選択肢」には値に「選択肢文」と

「遷移先 ID」が格納されている配列の要素の識別子を指定する.「遷移先 ID」には「選択肢文」を利用者が選択した際の遷移先のステップ ID の識別子を指定するが，指定した「遷移先 ID」が選択肢の要素に存在しなくてもよい．存在しない場合，生成するシナリオを終了させる.

表 2-1 テーブル形式の変換定義

項目		説明
API		対話入力用外部システムのデータを取得するための API を指定，スロットを利用可能.
データ構造		「テーブル」を指定.
マップ情報	発話文列番号	発話文が記述されている配列要素番号.
	選択肢 1~n 列番号	選択肢が記述されている配列要素番号. 選択肢 1～n を指定可能.
	ヘッダー行	テーブルにヘッダー行がある場合は「有」を，ない場合は「無」を指定.
修正		再入力用に生成シナリオを拡張する場合は「有」を，しない場合は「無」を指定，詳細は 2.3.4 節で説明.

表 2-2 オブジェクト形式の変換定義

項目		説明
API		対話入力用外部システムのデータを取得するための API を指定，スロットを利用可能.
データ構造タイプ		「オブジェクト」を指定.
マップ情報	ステップ ID	生成されるステップにつける ID の識別子.
	発話文	表示する発話文の識別子.
	選択肢	選択肢の識別子.
	選択肢文	選択肢内の発話文の識別子.
	遷移 ID	選択肢が選択されたときの遷移先の識別子.
修正		再入力用に生成シナリオを拡張する場合は「有」を，しない場合は「無」を指定，2.3.4 節で説明.

2.3.3 シナリオ生成方法

フロー連携方式は，変換定義の API の情報をもとに対話入力用外部システムからデータを取得し，変換定義の指定に従いシナリオを生成する．

図 2-3 (a)のテーブル形式のデータを取得し，図 2-4 の料理注文シナリオにフロー連携方式を適用する場合のシナリオの生成の流れを図 2-6 に示す．このとき，フロー連携用ステップを適用したシナリオおよびフロー連携用ステップの変換定義を図 2-6 の上部に示す．フロー連携用ステップ「注文内容選択」における対話入力用外部システムのデータ取得先は，変換定義の API「example.com/store?id=slot[店舗選択]」であり，前のステップである「店舗選択」ステップの選択結果データが参照される．もし， slot[店舗選択]の値が「ピザ店」の場合は対話入力用外部システムのピザ店のデータ，もし，slot[店舗選択]の値が「弁当店」の場合は弁当店のデータが取り込まれる．これにより 1 つのフロー連携用ステップで，対話入力用外部システムの複数の店舗の注文のシナリオに対応できる．

以後のシナリオの生成の流れは，slot[店舗選択]の値がピザ店の場合で，図 2-3 (a)のデータを取得したものとして説明する．まずステップを生成する．図 2-3 (a)のデータは 4 行*4 列の二次元配列であり，第 1 行はヘッダー行であり残りの 3 行が実データ行である．変換定義の発話文列番号に「1」，選択肢列番号に「2～4」が指定することで，取得データ（対話入力用外部システム店舗マスターのピザ店）の第 1 列目（すなわちヘッダーでいう「メニュー」）を「発話文」（図 2-6 内青枠），第 2~4 列目（すなわちヘッダーでいう「選択 1~3」を「選択肢」（図 2-6 内緑枠），行番号を「ステップ ID」とした「ステップ」を生成する．例えば，取得データの 2 行目からステップ ID が「2」，発話文「ピザを選択ください」，選択肢「チーズ，ミックス」というステップを生成する．これを行数文繰り返すが，ヘッダー行はステップの生成対象ではないため，変換定義で「ヘッダー行」を「有」とすることで 1 行目を飛ばす（図 2-6 内赤線）． 次に遷移を生成する．テーブル形式では，選択時に遷移先を分岐することはできず，すべてのステップが直列につながり，最終行のステップが完了すると生成したシナリオが終了する遷移を作成する．以上の方式を疑似プログラム言語で書いたものを，アルゴリズム 2-1 に示す．

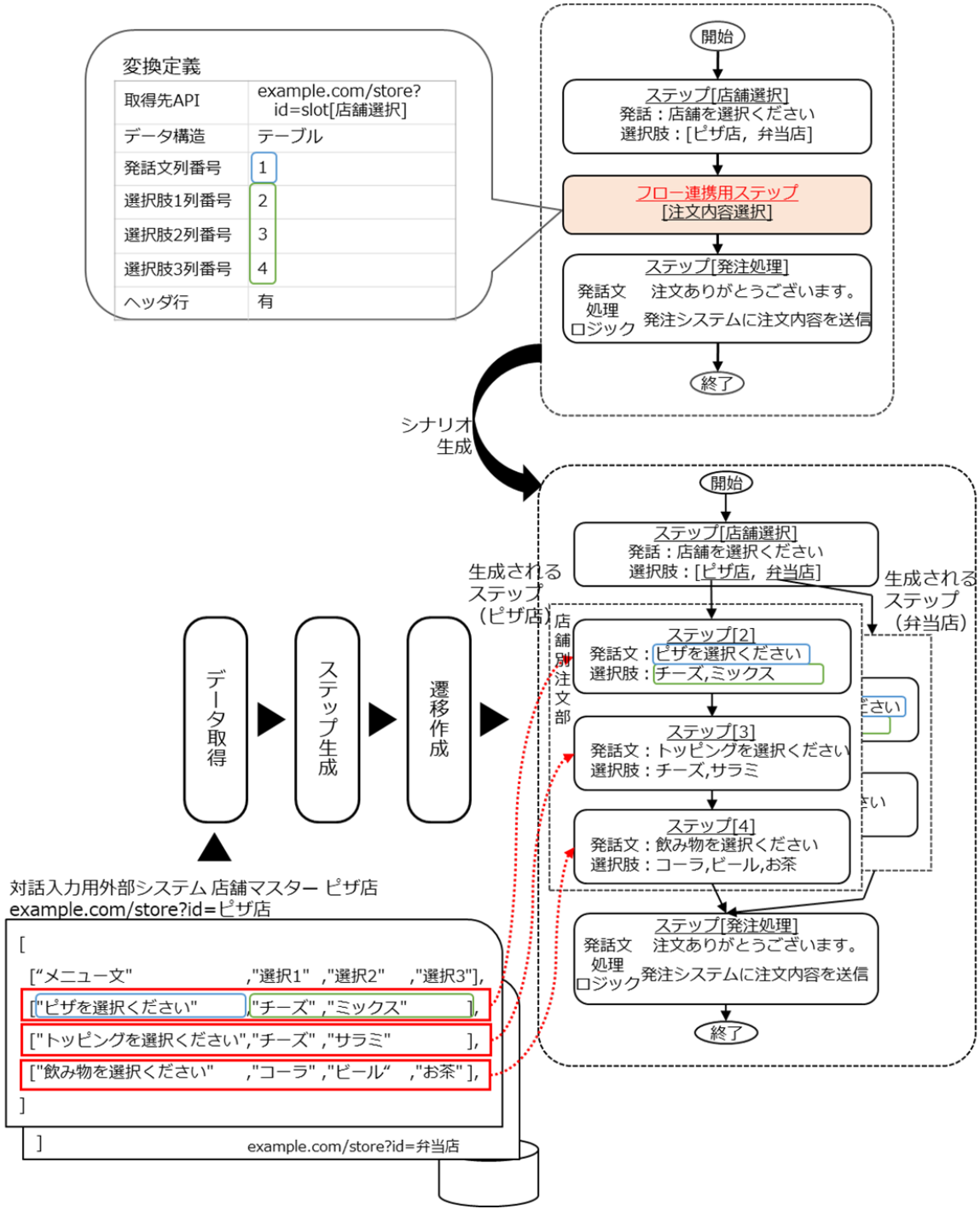

図 2-6　図 2-3 (a) に対する変換定義によるシナリオの生成例

アルゴリズム 2-1 テーブル形式のシナリオ生成処理

```
        Input：Conversion_Definition as D
01:     init step_list
02:     init transition_list
03:     # Get Data
04:     get_data = exising_system.get_data (D.API)
05:     # Generate steps
06:     if(D.Header_Row == "有") s = 2 else s = 1
07:     for i = s to get_data.length do
08:         id = i
09:         sp_text = get_data[i][D.Column_number_of_the_Speech_Text]
10:         add get_data[i][D.Column_number_of_the_Choice1~n] to choices
11:         add create_step(id, sp_text, choices) to step_list
12:     end do
13:     # Generate transitions
14:     for i = s to get_data.length do
15:         if(i+1 <= get_data.length) destination_id = i + 1
16:             else destination_id = "end"
17:         rule = true
18:         add create_transition(source_id=n, destination_id, rule) to transition_list
19:     end do
20:     scenario = create_scenario(step_list, transitions_list)
21:     return scenario
```

次に，オブジェクト形式のデータに，フロー連携方式を適用してシナリオを生成する流れを図 2-7 を例に説明する．このときの，フロー連携用ステップを適用したシナリオおよびフロー連携用ステップの変換定義を図 2-7 の上部に示すものとする．

図 2-6 と同様に，slot[店舗選択]の値が「ピザ店」とし，図 2-3 (b)のデータを取得するとして説明する．図 2-3 (b)のデータは 3 つの要素を持つ配列である．変換定義の「ステップ ID」に「メニュー名」，「発話文」に「メニュー文」，「選択肢」に「メニュー選

択」,「選択肢文」に「表示」を指定してあるので，対応する取得データの値からシナリオのステップを生成する（図 2-7 内赤線）．例えば，取得データの 1 つ目の要素から，ステップ ID に「ピザ選択」，発話文に取得データのメニュー文の値「ピザを選択ください」，ステップの選択肢として取得データの選択肢文の値である「チーズ」「ミックス」を持つステップが生成される．これを要素数分繰り返すことで 3 つのステップが生成される．

次に，図 2-7 ではステップ間の矢印で表されている遷移の情報を生成する．このとき，変換定義の「選択肢」「遷移 ID」に従い，選択肢ごとに，遷移元のステップの ID，遷移条件を「その選択肢を利用者が選択したとき」，遷移先 ID を「遷移 ID に指定されたステップの ID」とする．「ピザ選択」ステップの「ミックス」からは「次のメニュー」の値と同じ「ステップ ID」である「トッピング選択」ステップへの「遷移」,「チーズ」からは「次のメニュー」の値が「飲み物選択」のため，「飲み物選択」ステップへの遷移が作成される．「トッピング選択」ステップの「チーズ」「サラミ」からは「飲み物選択」ステップへの「遷移」を作成する．「飲み物選択」ステップの「コーラ」「ビール」「お茶」には，「次のメニュー」の値が存在しないため，シナリオを終了するための遷移を作成する（図 2-7 内の青線）．利用者が弁当店を選択した場合などで，取得データのオブジェクトの数（すなわち，生成されるステップの数）や選択肢配列のオブジェクトの数や遷移先 ID（すなわち，選択肢の数や遷移構造）が異なっていても，同様の方法で正しくシナリオを生成できる．

以上の方式を疑似プログラム言語で書いたものを，アルゴリズム 2-2 に示す．

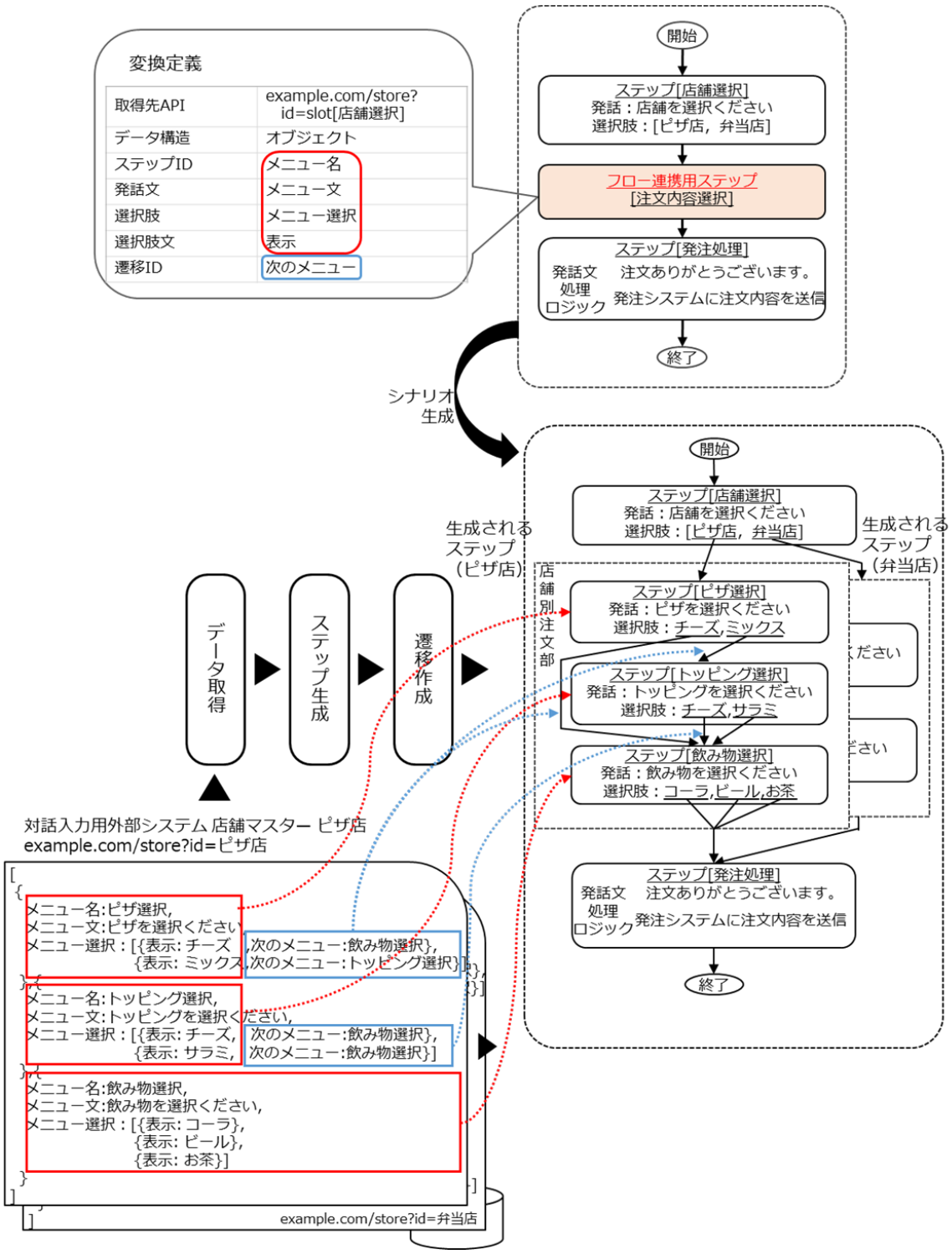

図 2-7　図 2-3 (b) に対する変換定義によるシナリオの生成例

アルゴリズム 2-2 オブジェクト形式のシナリオ生成処理

```
      Input : Conversion_Definition as D
01:   init step_list, transition_list
02:   # Get Data
03:   get_data = exising_system.get_data (D.API)
04:   # Generate steps
05:   for n = 1 to get_data.length do
06:      id = get_data[n][D.Step_ID]
07:      sp_text = get_data[n][D.Speech_Text]
08:      choice_text = get_data[n][D.Choices][[D.Choice_Text]]
09:      add create_step(id,sp_text, choice_text) to step_list
10:   end do
11:   # Generate transitions
12:   for n = 1 to get_data.length do
13:      init temp_transtion_list
14:      source_id = get_data[n][D.Step_ID]
15:      choices = get_data[n][D.Choices]
16:      for c = 1 to choices.length do
17:         destination_id = choices[c][D.Destination_ID]
18:         if destination_id
19:            rule = "slot[source_id] == choices[c][D.Choice_Text]"
20:            add create_transition(source_id,destination_id,rule)
                                                to temp_transtion_list
21:      end do
22:      if !temp_transtion_list
23:         rule = true
24:         add create_transition(source_id,"end",rule)
                                                to temp_transtion_list
25:      add temp_transtion_list to transition_list
26:   end do
27:   scenario = create_scenario(step_list, transitions_list)
28:   return scenario
```

2.3.4 変換定義の作成支援

フロー連携用ステップの作成時に，シナリオ作成ツールから変換定義に指定された API 出力結果をもとに対話入力用外部システムからデータを実際に取得することで, 以下の手順で変換定義のデータ構造およびマッピング情報の入力を支援する.

（1） 取得データの各要素のデータ構造が配列の場合は「テーブル」として，連想配列の場合は「オブジェクト」として，変換定義の「データ構造」を自動的に設定する.

（2） マッピング情報の各パラメータの入力をデータ構造により以下のように支援する.

データ構造がテーブル形式の場合：「発話文列番号」「選択肢 1~10 列番号」に，取得データの列番号の一覧をプルダウン形式で表示し, シナリオの作成者が選択できるようにする.

データ構造がオブジェクト形式の場合：データの意味は機械的にはわからないため, 各要素において, 値が配列であるかそうでないかというデータの構造的要件に基づき指定候補を決定する. 値が配列ではない要素が一つだけの場合や, 配列である要素が一つだけの場合は，それらを自動的にパラメータとして設定する. 値が配列ではない要素が複数ある場合や, 配列である要素が複数ある場合はそれらを指定候補としてプルダウン形式で表示し，シナリオ作成者が選択できるようにする. 各パラメータの指定要件は以下のとおりである

- 「ステップ ID」,「発話文」: 取得データで値が配列ではない要素の識別子
- 「選択肢」: 値が配列である要素の識別子
- 「選択文」,「遷移先 ID」: 指定された「選択肢」要素内の識別子

例えば，図 2-3 (b)のオブジェクト形式のデータを対話入力用外部システムのデータとして, API により取得したとする. この際のシナリオ作成ツール上での変換定義の入力支援を利用した発話文の指定例を図 2-8 に示す. 図 2-3 (b)のデータは連想配列であるため，変換定義の「データ構造」に「オブジェクト」が自動で指定される.「マッピング情報」の「ステップ ID」と「発話文」には，図 2-3 (b)のデータの要素のうち，値が配列ではない要素の識別子は「メニュー名」と「メニュー文」であるため，それらが指定候補としてプルダウン形式で表示され(図 2-8 内赤枠), シナリオ作成者が適切な識別子である「メニュー文」を選択することができる(図 2-8 内青枠).「選択肢」は，値が

配列の要素の識別子が「メニュー選択」のみのため自動で設定される．「遷移先 ID」と「選択肢文」には，「選択肢」に「メニュー選択」が指定されることで，その「メニュー選択」内の要素の識別子である「表示，「次のメニュー」が候補として表示され，シナリオ作成者が適切な識別子（「選択肢文」に「表示」，「遷移先」に「次のメニュー」）を選択することができる．

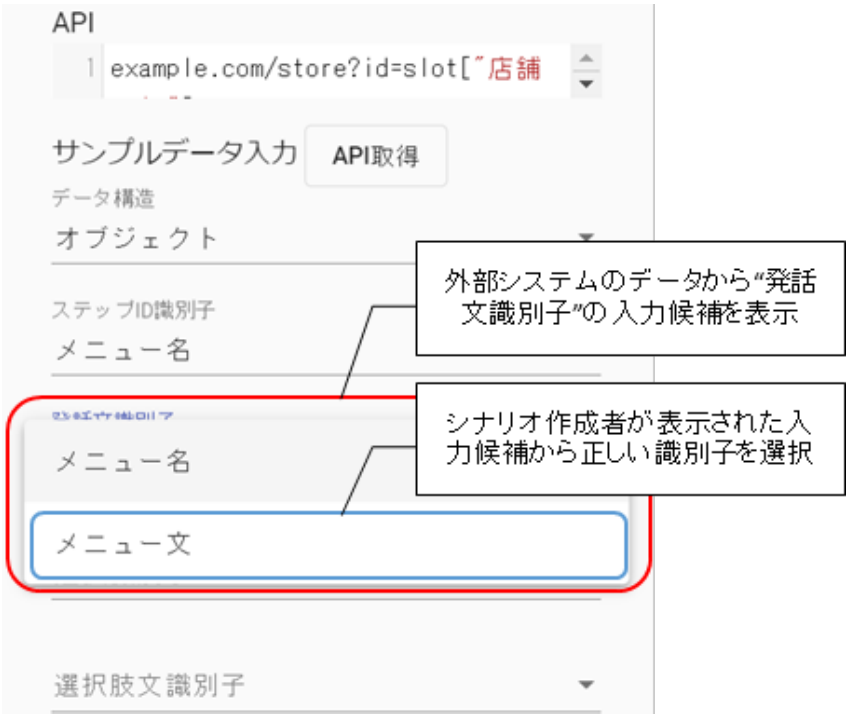

図 2-8 シナリオ作成ツールによる変換定義の作成支援例

2.3.5 利用者入力の確認修正ステップの自動生成

生成シナリオでステップごとに既回答を表示し修正するかどうか利用者に聞くことができるようにシナリオを自動的に拡張することで，容易に対話結果を修正できるようにする．具体的には，変換定義の「修正」が「有効」の場合，フロー連携ステップの再実行時に，過去（修正前）の対話結果のスロットを引き継ぎ，図 2-9 に示すシナリオのように，各ステップ実行前に「書込予定のスロットに既に値があったらその値を利用者に提示し修正するかを確認するステップ」を自動的に追加する(図 2-9 内赤枠)．さらに，生成シナリオの最後で再実行時に利用されなかったスロットを削除するステップを追加する(図 2-9 内青枠)．このステップにより，再対話時に遷移が変更され実行されるステップ・書込スロットが変わった場合に，修正前の対話内容がスロットに残るのを防ぐ．

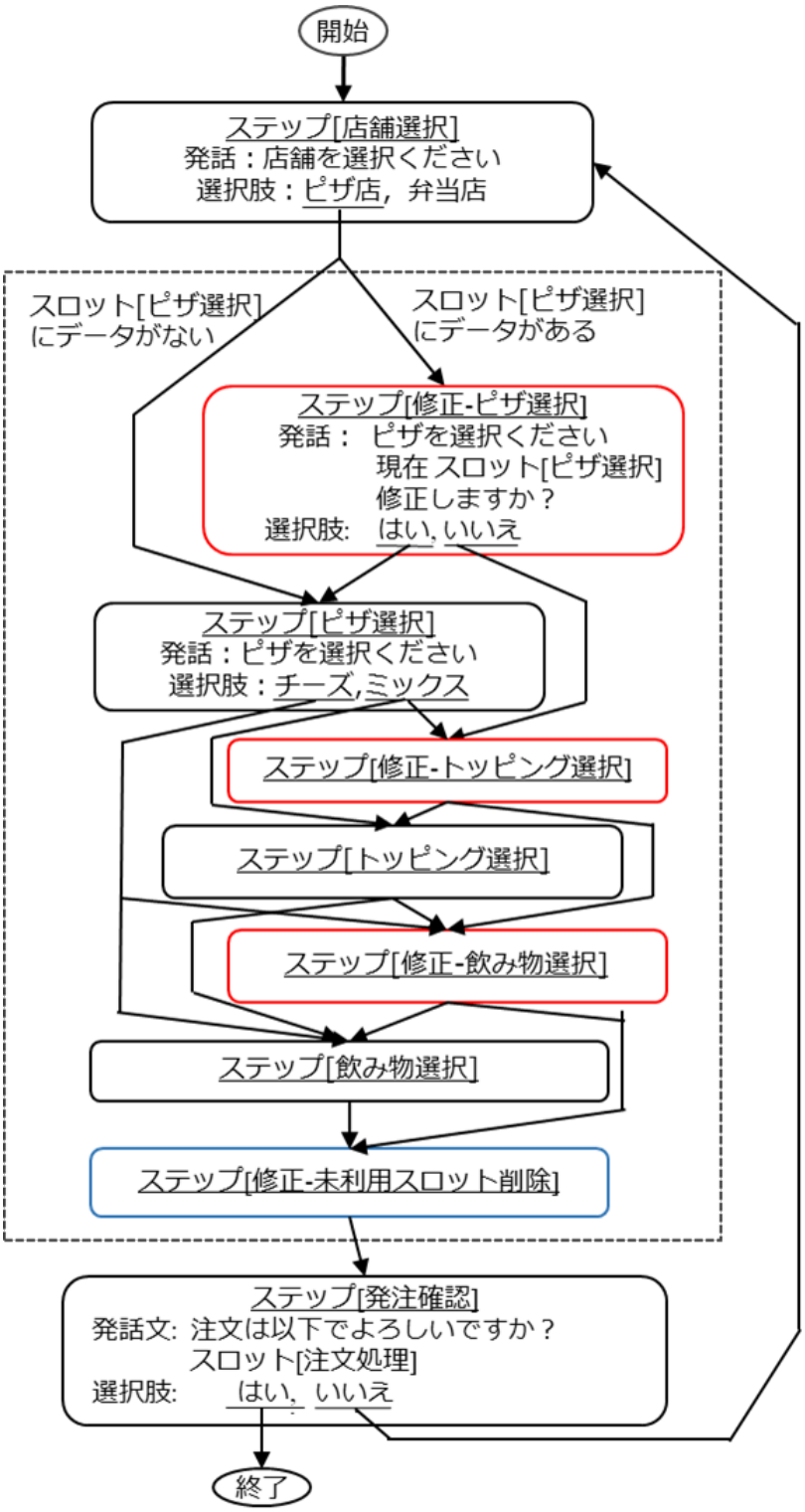

図 2-9 修正時の生成シナリオの拡張

2.4　比較評価結果および考察

2.4.1 評価方法

評価実験として，提案のフロー連携方式を利用した場合と，従来のパラメータ連携方式を利用した場合のシナリオの作成時間を比較した. 連携対象となる既存の対話入力用外部システムが存在する実稼働している 2 つのシナリオを実験対象とした. 評価に使用したシナリオの規模を表 2-3 に示す.

表 2-3 評価に使用したシナリオの規模

シナリオ名	生成したステップ数	データ構造
A	100	オブジェクト
B	17	テーブル

生成ステップ数は，フロー連携方式で，対話入力用外部システムからシナリオを生成した際の生成シナリオに含まれるステップ数の合計である．フロー連携用ステップは，シナリオ A・B ともに 1 ステップである．フロー連携用ステップで生成されるステップは，シナリオ A では 8 つに分岐している．一方，シナリオ B では，5 つに分岐している．

シナリオを作成したシナリオ作成者は，シナリオ A ではフロー連携方式・パラメータ連携方式で異なる担当者が対応した. 両方式の担当者ともに大きな能力差はなくシナリオ作成に初めて触れる初心者であり, 対話入力用外部システムに対する理解も同程度である．シナリオ B は，シナリオの作成経験があり対話入力用外部システムにも精通している SE（System Engineer）が，フロー連携方式，パラメータ連携方式の順に作成した.

作成に要した時間は，以下の方法で算出した.

・フロー連携方式を利用の場合：

シナリオ作成者の作業記録から, フロー連携方式を利用した変換定義の定義時間を測定した. この作成時間には, 利用する対話入力用外部システムの仕様（API の使い方や，取得データの構造）の把握や，フロー連携用ステップのテスト（API との疎通確認やシ

ナリオの生成テストおよびテスト結果のエビデンスの記録）の時間も含んでいる．

・パラメータ連携方式を利用の場合：

シナリオ作成者がシナリオ作成ツールを使って，シナリオのステップやステップ間の遷移を作成し，ステップの発話文や選択肢といったパラメータは対話入力用外部システムのデータを参照した．この作成時間には，シナリオの記述方式の調査，パラメータに利用する対話入力用外部システムの仕様の把握や，作成したシナリオのテスト（作成したシナリオが意図通りに動くかの対話テストの作成およびテスト結果のエビデンスの記録）時間が含まれている．

両方式ともに，提示された仕様どおりにチャットボットと利用者が対話できる同じシナリオが作成できた．

2.4.2 評価結果

それぞれの方式でシナリオの作成に要した時間を比較した結果を表 2-4 に示す．シナリオ A は，それぞれの方式で異なる担当者が担当しており，単純比較はできないが，81%の削減となり，フロー連携方式により一定の削減ができたと考えられる．シナリオ B では，両方式に同一の担当者が担当しており，フロー連携方式でのシナリオ作成において対話入力用外部システムに関する理解をしたうえでパラメータ連携方式の作業に着手している．表 2-4 のデータは，フロー連携方式の作業時間は対話入力用外部システムの理解時間を含むのに対し，パラメータ連携方式の作業時間は対話入力用外部システムの理解時間を含まないにも関わらず 53%削減できた．この実験から，シナリオの作成に要する時間は，「フロー連携方式を利用した場合」では，「パラメータ連携方式を利用した場合」と比べて，シナリオ A，シナリオ B ともに 50%以上削減されたと言える．シナリオ B に比べてシナリオ A の削減率が高いのは，シナリオ A が，シナリオ B に比べて複雑で，フロー連携方式により自動生成されるステップが，多かったためと考えられる．

また，フロー連携方式では，ステップ内部の発話文や選択肢，処理ロジックなどの項目に加えて，遷移やステップも含めたシナリオ自体を作成でき，パラメータ連携方式に比べて，より多くの対話入力用外部システムのデータを利用できる．これによりパラメ

ータ連携方式に比べて，対話入力用外部システムのデータとシナリオの定義情報の重複管理を削減できることを確認した．なお，実際に業務担当者が対話入力用外部システムのデータを更新することで容易にシナリオを更新できることを確認し，業務担当者からも本方式は有用であるとのコメントを得られた．

表 2-4 シナリオ作成時間の結果比較

シナリオ名	フロー連携方式	パラメータ連携方式
A	13 時間	68 時間
B	7 時間	15 時間

2.5 結言

本章では，対話入力用外部システムにチャットボットのシナリオのステップや遷移を生成できる情報があり，その情報を取得できることを前提に，シナリオを生成する方式を提案した．対話入力用外部システムのデータからシナリオを生成するために，対話入力用外部システムのデータとシナリオの構造を対応させる変換定義と，それにより定義される対話入力用外部システムのデータからシナリオを自動生成するシナリオ生成方法を提案し，試行したチャットボットの適用において，提案方式は手動でシナリオを作るのに比べて，シナリオの作成時間を 5 割以上削減できる見通しを得た．また，提案方式は，ステップ内部の項目に加えて，遷移やステップも含めたシナリオ自体を作成でき，従来方式に比べて，対話入力用外部システムのデータとシナリオの定義情報の重複管理を削減できることを確認した．

なお，現時点では，評価人数，評価対象シナリオ数が少なく，評価結果の確度は不十分であり，今後，様々なユースケースのシナリオにおいて，さらなる評価の充実が必要である．さらに，対応できる対話入力用外部システムの種類についても，代表的なデータ構造形式に対応してはいるものの，対話入力用外部システムのすべてのデータ構造形式に対応しておらず，容易に対応形式を増やす仕組みなどの検討が必要である．また，チャットボットの並行導入時，チャットボットで代替したいタスクの情報が，企業によってはテキストデータなどの非構造化データで管理されていることがある．これら非構造データからシナリオを生成することが求められる．例えば，オペレータ用の電話対応

の手順書が自然言語のテキストデータで管理されていた場合, 手順書のテキストデータを自然言語処理技術[88], [89]などで解析することで, 利用者に聞くべき質問を抽出してステップを生成し，手順の「年齢が 18 歳以下だったら」などの条件の記述から，選択肢や遷移を生成することが考えられる．さらに，特定のドメインの利用が多く想定される場合は，背景知識の利用等により更なる記述の容易化実現の可能性がある．

第3章
動作条件によるシナリオのテスト方式

3.1　緒言

本章では, シナリオのテストとデバックの効率化に向けて, 対話時の仕様の動作から, シナリオが誤っている可能性がある箇所を機械的に確認し, シナリオの作成者に提示する方式を提案する．最近のチャットボットでは，業務担当者がシナリオを開発することが増えてきており，シナリオの開発の簡単さや，チャットボットの提供までのスピードが重視されている．しかし，業務担当者がシナリオをテスト，デバックするには，テスト方法の簡単化が不十分である．時間をかけて，すべての作成不良を発見・解消するよりも，基本的な動作のみをテストし，問題ないと業務担当者が判断すれば，チャットボットを提供してしまうことが多い．この方法では，運用後に作成不良が顕在化しチャットボットに異常な動作が起きることがあり，システムが停止するなど問題となる．このため, チャットボットの提供までのスピードや開発の簡単さを優先しながらできる限り作成不良を発見できる方法が求められている．

テストで一般的によく用いられる方法としては, シナリオ作成者がシナリオを作成しながら，適時，実際に対話型 UI を利用してチャットボットと利用者の対話を模擬し，その対話結果からシナリオが仕様通りに動くかを確認する方法がある．しかし，この方法は，シナリオの異常動作を発見するのに対話を繰り返すことが多く，シナリオ作成者による模擬対話に時間を要する. チャットボットの提供スピードやシナリオの作成者の負担軽減を優先した結果, シナリオの異常動作を発見するのに必要なすべての模擬対話を実施せず，作成不良の見逃しにつながる．

また，シナリオの作成者が，利用者による一連の想定発話文と想定発話文に対する期待されるチャットボットの返答結果のリストを事前に指定することでチャットボットを自動でテストする方式も提案されている[55], [56]（以後，この方法を自動テスト方式と呼ぶ）．この方法は，シナリオの作成者が正しく想定発話分と返答結果を指定できれば，シナリオにあるすべての作成不良を発見できる．しかし，チャットボットやプログラミングの知識が浅い業務担当者などが, すべての想定発話文と期待される返答結果の

リストを正しく作成しなければならず，シナリオの更新の際にそれらのリストも正しく更新しなければならない．これは，シナリオ作成者の負担となるため適用が難しい．

そこで，本章では，従来のテスト方法よりも，シナリオ作成者によるテストに必要な時間をできる限り減らしながら，作成不良の見逃しを削減することを目的とし，シナリオ作成者が対話時の仕様の動作を動作条件としてシナリオに記述できるようにすることで，シナリオを機械的にテストする方式を提案する．

以下，第 3.2 節でシナリオの作成不良とテスト方法の現状とその問題点を説明し，第 3.3 節で提案する動作条件によるテスト方式を説明する．最後に，第 3.4 節で実際のチャットボットのシナリオ作成を例に，提案方式の評価を行う．

3.2　シナリオのテストの現状

3.2.1　シナリオの作成不良

シナリオを作成する際，対話の流れを誤って指定するなどの作成不良が発生する．例えば，遷移条件を誤って記述し意図しない遷移となる，ステップで書き込むスロットを仕様と誤って指定するなどである．

シナリオ作成時に発生する主な作成不良を表 3-1 に示す．

表 3-1 シナリオ作成時に発生する主な作成不良

分類	作成不良	説明
スロット指定ミス	読込スロット指定ミス	発話文や選択肢・遷移条件で利用するスロットの指定誤り
	書込スロット指定ミス	ステップの結果を書き込むスロットの指定誤り
遷移指定ミス	遷移元指定ミス	遷移元 ID の指定誤り
	遷移先指定ミス	遷移先 ID の指定誤り
	遷移条件指定ミス	遷移条件の指定誤り
静的指定ミス	発話文指定ミス	静的な発話文の指定誤り
	選択肢指定ミス	静的な選択文の指定誤り
	処理ロジック指定ミス	静的な処理ロジックの指定誤り

読込スロット指定ミスは，ステップの読込スロット指定を誤り，想定と異なるスロットを読み込むことである．この結果として，書き込んでいないスロット，存在しないスロットを読み込む読込違反などの異常動作が発生する．

書込スロット指定ミスは，ステップの書込スロット指定を誤り，想定と異なるスロットに書き込むことである．この結果として，意図しないスロットの上書き，読込違反などの異常動作が発生する．

遷移元指定ミス・遷移先指定ミス，遷移条件指定ミスは，遷移作成時に遷移元や遷移先，遷移条件の指定を誤ることである．この結果として，ステップの意図しない実行，シナリオが絶対に終わらない無限ループなどの異常動作が発生する．

静的指定ミスは，発話文や選択肢，処理ロジックの計算処理の静的な文字列を誤ることである．この結果として，チャットボットの利用者に対して意図しない発話文などが表示される．

この他の作成不良としては，遂行するタスクの仕様では必要であった機能がシナリオ上から漏れてしまう機能実装漏れなどがある．例えば，利用者に住所を質問しなければならい仕様であったのに，シナリオに住所を聞くステップを作らなかった場合などがあてはまる．

具体的にシナリオの作成不良の例を図 3-1，図 3-2 に示す．図 3-1 は，読込スロット指定ミスの作成不良により読込違反の異常動作が起こるシナリオである．このシナリオは，料理の注文シナリオであり，チャットボットは利用者が注文したい店舗を確認し，その店に応じたメニューを受け付ける．ピザ選択ルート（データ初期化 ―> メニュー ―> ピザメニュー ―> ピザオプション選択 ―> 注文確認 ―> 注文ステップ）は「ピザオプション選択」ステップでオプションを指定するため「option1」スロットに書き込む．しかし，弁当選択ルート（データ初期化 ―> メニュー ―> 弁当メニュー ―> 注文確認 ―> 注文ステップ）では，オプション選択が存在しないため，「option1」スロットへの書き込みが存在しない．この結果，弁当選択ルートでは，注文確認ステップ実行時に「option1」スロットを表示しようとして「option1」スロットに値がなく，読込違反が発生する．ここで，ルートとは，利用者との 1 対話（開始から終了まで）の実行ステップの流れ（経路）を表す．

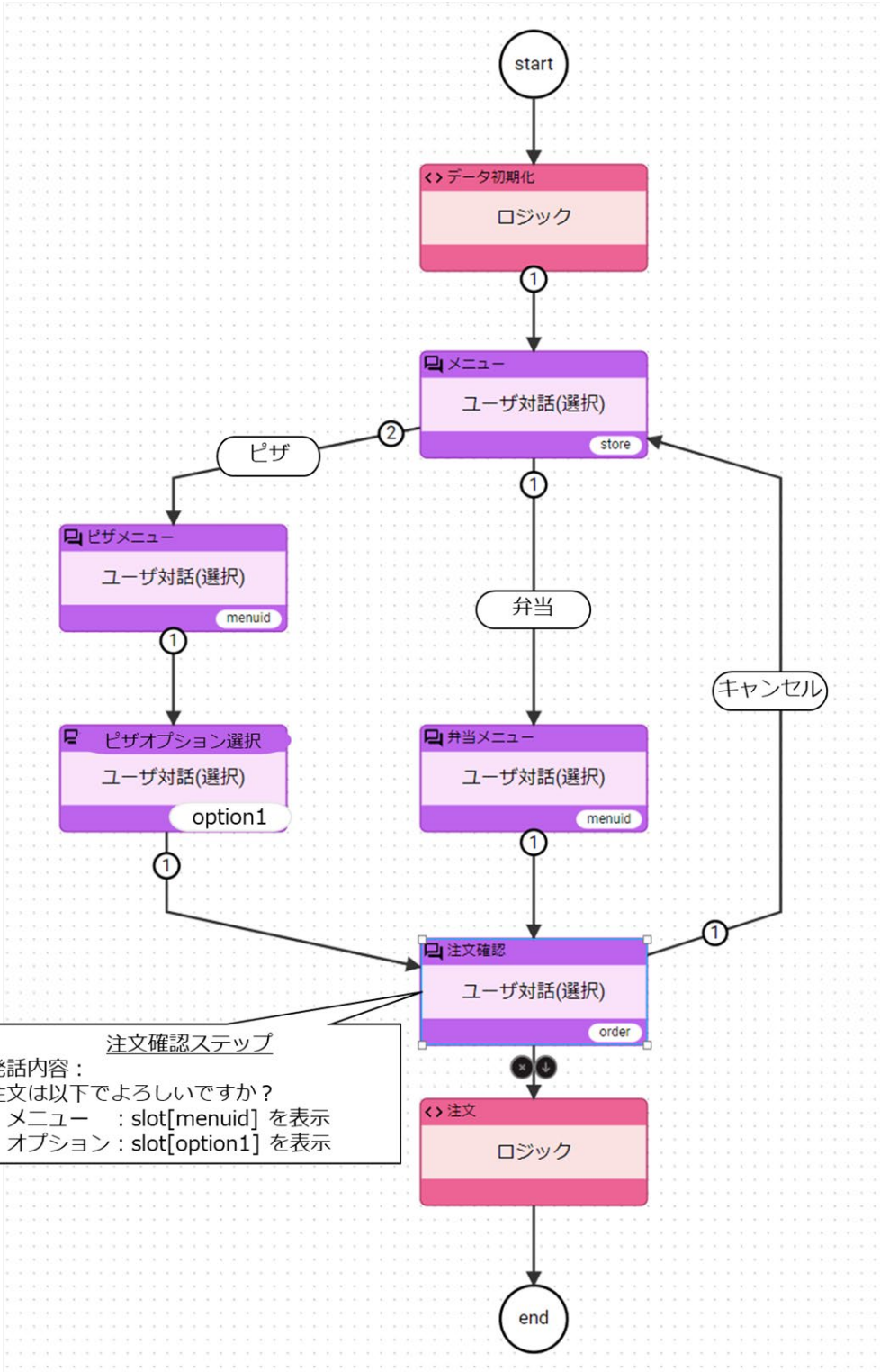

図 3-1 読込スロット指定ミスのシナリオ例

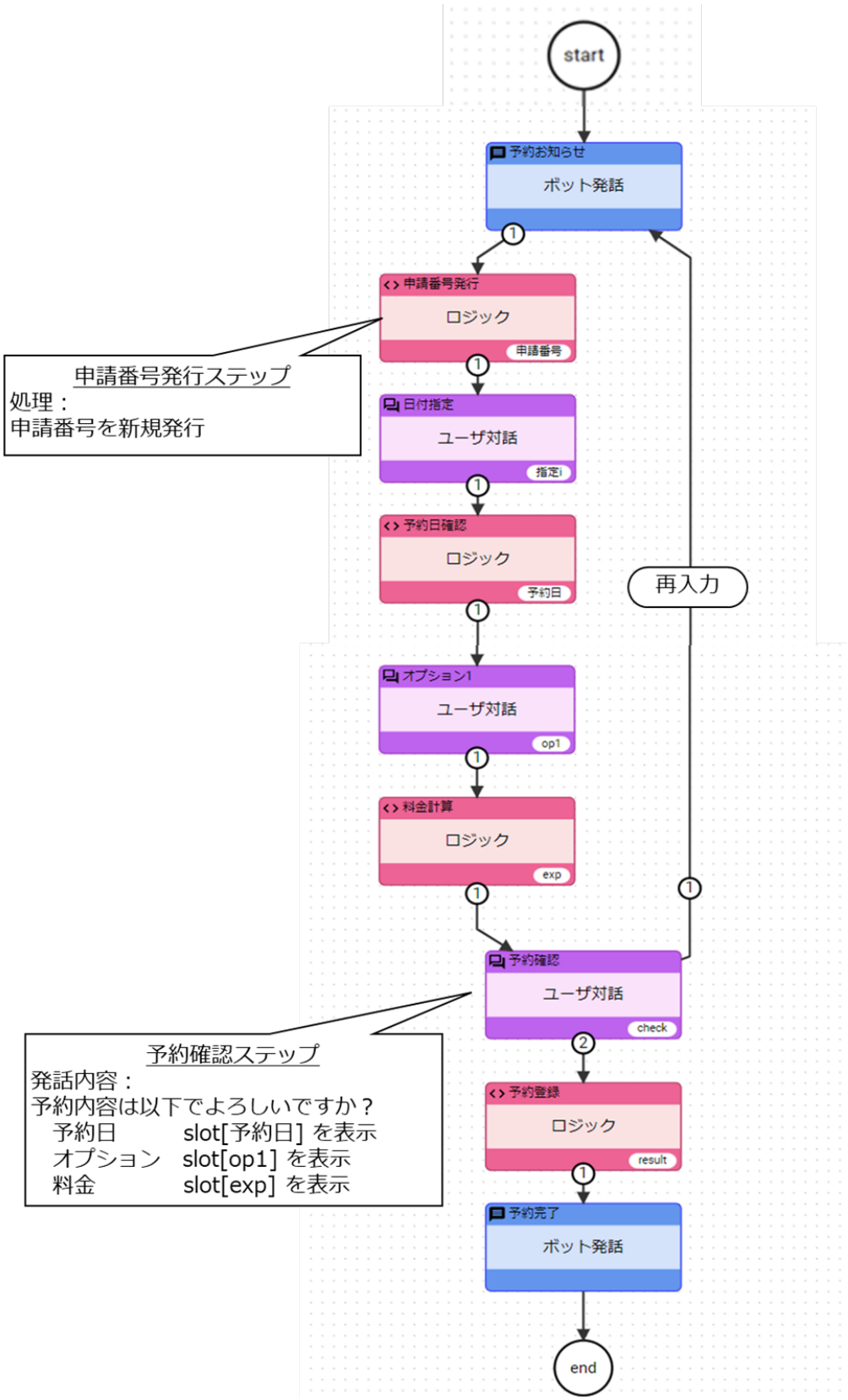

図 3-2　遷移先指定ミスのシナリオ例

図 3-2 は，遷移先指定ミスによる作成不良により，ステップの意図しない実行が起こるシナリオである．このシナリオは，施設予約申請のシナリオであり，「申請番号発行」ステップで申請番号を発行する．この申請番号は，申請単位で発行されるものである．その後，予約日やオプションを利用者が指定後，料金を計算し，予約確認ステップで，申請内容を利用者に確認する．この際に，利用者の選択に誤りがあった場合，「再入力」を選択することで，申請内容を再入力することができる．しかし，この再入力の遷移先（予約確認ステップから遷移条件「再入力」により遷移する先）が「予約お知らせ」ステップになっており，その後「申請番号発行」ステップが実行される．この結果，再入力のたびに「申請番号発行」ステップが再実行されて，申請番号が多重発行される．特に，この異常動作は，申請番号が再発行されるだけで利用者との対話は正常に進んでいるように見え，かつループ（前のステップへの遷移）も含めて確認する必要があるため，チャットボットと利用者の対話の模擬結果から確認する方法では発見しにくい．

3.2.2 テスト方法とその課題

作成したシナリオは，前述の作成不良がなく，シナリオの作成者の意図通りに動くかをテストし，何らかの問題があればそれをデバッグする．テストでは，シナリオにおける以下の 2 つの観点を確認する．

・シナリオに記述された対話内容が正しいか

シナリオでチャットボットが利用者に発話するテキストがシナリオの作成者の意図通りに利用者に表示されるかを確認する．この観点のテストでは，シナリオを構成するステップごとに静的指定ミスに起因する異常動作が発生しないかを確認することが目的となる．

・シナリオが構造的に正しいか

シナリオの作成者が意図した対話の流れを，シナリオ上で正しく構造化できていることを確認する．この観点のテストでは，複数ステップを横断する構造の正しさを確認する．具体的には，3.2.1 節で説明したスロット指定ミス，遷移指定ミス（以後，この 2 つをまとめて構造的な作成不良と呼ぶ）に起因する異常動作が発生しないかを確認することが目的となる．

構造的な作成不良がシナリオに存在すると，対話時にシナリオの複数の要素（ステッ

プや遷移，スロット）が関係した結果，ステップの実行，スロットの読み書き，順序関係が誤って動作し，結果としてチャットボットが意図しない返答をする．テストでは，発生した動作が異常動作なのかをチャットボットの返答などから判定する．このとき，どのような対話の経路（以後，ルートと呼ぶ）により，その動作が発生したかは，異常動作の判定に影響しない．例えば，図 3-1 のシナリオで，スロットに書込んでいないのに読み込む動作，図 3-2 のシナリオで「申請番号発行」が複数回実行される動作が発生することが分かれば，発生するルートは関係なく，シナリオに作成不良があることをシナリオ作成者は判定できる．

これらの観点をテストするために一般的に用いられる方法として，シナリオの作成者がシナリオを作成しながら，随時対話を模擬し，シナリオに異常動作がないか確認する（以後この方法を模擬対話方式と呼ぶ）． 模擬対話方式では，シナリオ作成者がシナリオのすべてのルートを満たすように利用者の発話を模擬して対話を実施し，発話に対するチャットボットからの返答内容をもとに上記 2 つの観点を確認する．

表 3-2 図 3-2 のシナリオに対して
模擬対話方式によるテストを実行する際のテストケースと発話内容一覧

#	テストケース 1 (ピザルート)	テストケース 2 (弁当ルート)	テストケース 3 (ピザ-キャンセル-弁当ルート)	テストケース 4 (弁当-キャンセル-ピザルート)
1	ピザ	弁当	ピザ	弁当
2	ミックスピザ	唐揚げ弁当	ミックスピザ	唐揚げ弁当
3	チーズ	はい	チーズ	キャンセル
4	はい		キャンセル	ピザ
5			弁当	ミックスピザ
6			唐揚げ弁当	チーズ
7			はい	はい

図 3-2 のシナリオを例に，模擬対話方式によるテストの様子を説明する．このときのテストケースとそのケースを実行するための利用者を模擬した発話内容の一覧を表 3-2 に示す．テストケースは 4 つあり，テストケース 1 ではピザルート，テストケース 2 では弁当ルート，テストケース 3 は一度ピザのルートで料理を選択後(#1-#3)，注文確認ステップで注文をキャンセルして (#4)，弁当のルートで注文し直す(#5-7)ルート，テストケース 4 ではテストケース 3 と反対(弁当選択後キャンセルしてピザを選択し直す)を通るように発話し，結果を確認する．テストケース 2 を実際にチャットボットに対して発話すると図 3-3 に示すように，弁当選択後に読込違反の異常動作が発生し，その原因となる動作を探ると読込違反が発生していることがわかり，図 3-2 のシナリオに作成不良が存在することがわかる．このように模擬対話方式では，シナリオの作成者が，シナリオがとりうるすべてのルートを進むように発話し，ルートごとにチャットボットの返答から動作を確認する．

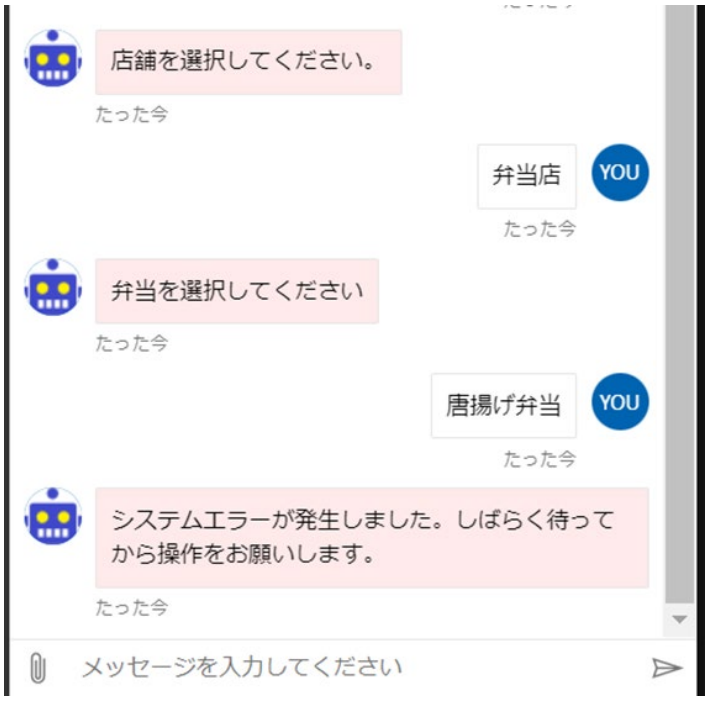

図 3-3 模擬対話方式によるテスト実行例

最近では，業務担当者がシナリオの作成を担当することが増えてきた結果，チャットボットの開発効率化のため，業務担当者がシナリオをテスト，デバックできることが求められている．しかし，チャットボット開発の知識を持たない業務担当者がシナリオを

テスト，デバックするのは以下の点から難しい．

前述の手順において，シナリオに作成不良がないかを確認するためのルートの数，模擬対話の回数は，シナリオの複雑さ（ステップの数や戻る遷移などによるループの数）により多くなる．シナリオに記述された対話内容が正しいかのテストは，シナリオを構成するステップを 1 回ずつ発話するように対話すれば確認できるが，シナリオが構造的に正しいかテストするためには，シナリオのすべてのルートを網羅的に対話する必要がある．業務担当者では，シナリオの構造をもとにすべてのルートを洗い出し，網羅的に対話するのは時間を必要とし，対話が足りず作成不良の見逃しが起こりやすく，シナリオの作成者の負担につながる．さらに，作成不良が発生した際に，シナリオの複数の要素が絡む作成不良である場合，作成不良の個所を見つけるために，何度も対話の繰り返す必要がある．特に業務担当者では，シナリオの作成経験が少ないことが多く，作成不良の個所の特定に時間がかかり，迅速化の妨げとなる．

3.3　動作条件によるシナリオのテスト方式

3.3.1　提案方式の概要

前節の問題に対して，構造的な作成不良が発生させるシナリオの異常動作に着目し，対話時の仕様の動作から，シナリオの誤っている可能性がある箇所を機械的にテストする方式を提案する．シナリオを機械的にテストするには，対象のシナリオで対話し動作を起こすための入力値と，入力値に対して対話時に起きた動作が異常であるか判定する条件が必要になる．3.2.1 節で説明したように，従来のテスト方式では，すべてのルートを通る利用者の発話文を入力値として，入力値毎のチャットボットの返答結果を条件とするが，これはシナリオ作成者の手間がかかり負担軽減にならない．このため，提案方式では，シナリオの構造的な作成不良がない場合の動作（スロットの読込違反がないなどの正しい動作）の条件を定義するシナリオ向きの動作条件の定義方法と，利用者の発話文なしでシナリオを動作させ，動作条件をもとに異常を判定する方法を定める．

提案方式では，シナリオに構造的な作成不良がない場合の動作（正しい動作）の条件を動作条件としてシナリオ作成者がシナリオ作成と並行して定義できるようにする．動作条件の詳細は 3.3.2 節で説明するが，入力値毎のチャットボットの返答結果ではなく，構造的な作成不良が起こす動作を条件にすることで，動作条件は入力値や対話時のルー

トに関係なくシナリオに共通に定義できる．これにより，入力値毎に判定する条件を指定する必要をなくし，条件の指定を効率化する．また，シナリオの作成者がすべての動作条件を作成するのは，作成ミスや時間が掛かる問題がある．そこで，すべてのシナリオにデフォルトで作成される共通動作条件とシナリオのユースケースに応じてシナリオの作成者が作成するカスタム動作条件に分けることで，シナリオ作成者がすべての動作条件を作る必要をなくす．

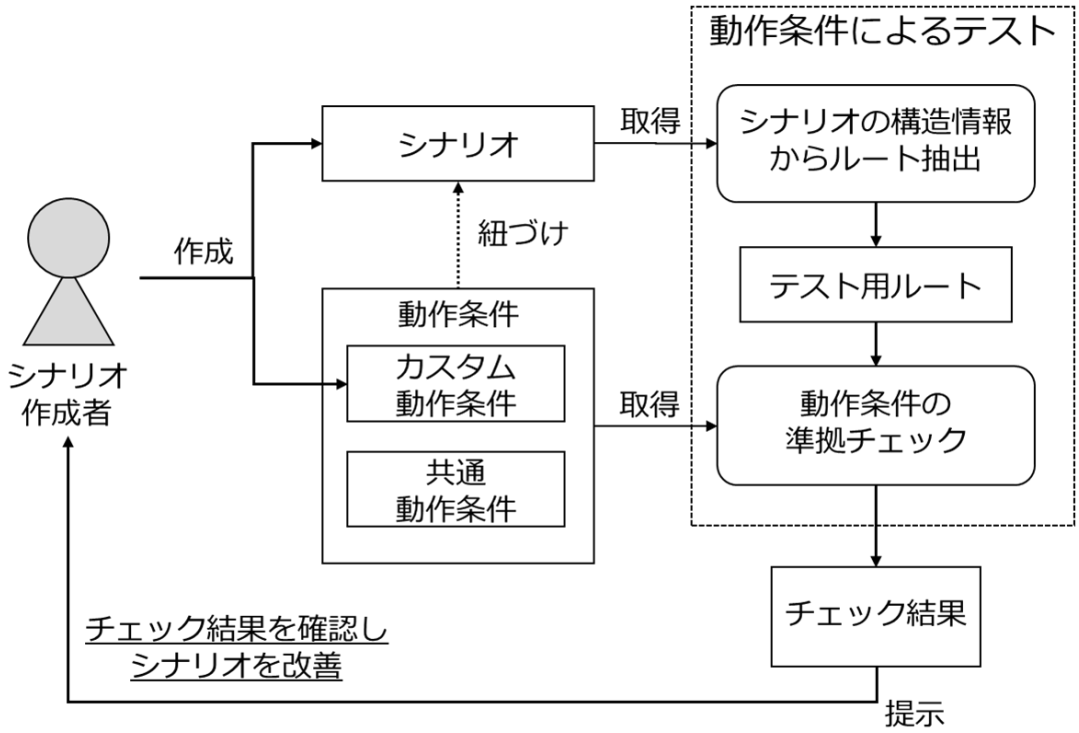

図 3-4 提案方式のモデル

図 3-4 に示すように，シナリオに紐づけられた動作条件とシナリオの構造情報から全ルートを抽出し，機械的にシナリオをテストする．チャットボットは本来利用者の発話により対話が進み動作することから，シナリオ作成者がすべてのルートで対話できるように利用者の想定発話内容を指定する必要があるが，これは，前述したように難しい．そこで，構造的な作成不良を起こす動作や動作条件の判定に利用者の発話内容やチャットボットの返答結果は関係ないことに着目する．提案方式では，実際にチャットボットと対話せずに，シナリオ上の構造情報からすべてのルートを網羅的に抽出する．抽出し

たルート毎に実行ステップや遷移のシナリオ上の定義情報のみから，起こる動作を確認し，指定された動作条件に準拠しているかをチェックする．これにより，シナリオ作成者が利用者の想定発話を指定しなくても，機械的にシナリオのすべてのルートが動作条件に準拠しているかを確認できるようにする．提案方式によるチェックの結果，動作条件に準拠しない動作が見つかった場合は，作成不良の可能性として，違反した動作条件，ルート，発生箇所をシナリオ作成者に提示し，修正を促す．

さらに，シナリオ作成者による動作条件の作成を容易にする．具体的には，チャットボットシステムでは，利用頻度が高い一定のユースケースのシナリオをテンプレートとして提供している．これに動作条件を事前に作成し紐づけておくことで，それらのテンプレートをもとにシナリオを作成した場合，動作条件を作成する必要を減らす．また，提案方式による動作確認で動作異常が見つかった場合に，それが正常な動作だった場合，動作異常の提示内容からその動作条件を無効化できるようにすることで，次回のテスト以降，動作異常と判定されなくする．

動作条件定義については 3.3.2 節で，動作条件によるテストの実行について 3.3.3 節で，動作条件の作成支援について 3.3.4 節でそれぞれ詳しく説明する．

3.3.2 動作条件定義

動作条件は，定義 3-1 に示すように，シナリオのどこに対する動作条件なのかを指定する「対象」，対象の対話時の仕様の動作を指定する「条件」の組みで定義する．「対象」は，「要素」と「範囲」から成り，「要素」は，動作条件の対象となるシナリオの構成要素を表し，「範囲」は動作条件の対象が，特定の要素なのか，すべての要素なのかを表す．「条件」は，「対象」がどのような動作や状態のときに動作条件に準拠しているかを表す．「条件」では，構造的な作成不良により起こる動作を異常と判定するように実行回数，読込回数，書込回数，順序条件のいずれかの条件から指定することができる．

動作条件の作成例を表 3-3 に示す．#1 の動作条件は，要素に「スロット」が，範囲に全体が指定されているため，シナリオ上のすべてのスロットが対象となる．条件に「順序条件（書込－＞読込）」が指定されており，シナリオのすべてのルートでスロットを読み込む前に書き込みがあることが条件となる．#2 は，ステップの多重実行を発見するための動作条件である．要素にステップが，範囲に「特定（ステップ X)」が指定さ

れているため，シナリオ内のステップ ID が X のステップを対象としている．条件が「実行回数（必ず 1 回）」であるため，シナリオ上のいかなるルートでも対象が 1 回実行されていることが条件となる．このため，#2 の動作条件では，シナリオ上のいかなるルートでも X ステップは必ず 1 回実行されていることを条件とすることで，2 回実行されるなどした場合は異常動作になる．#3 は，スロットの上書きを発見するための動作条件である．要素にスロットが，範囲に全体が指定されているため，シナリオのすべてのスロットを対象としている．条件は書込回数（1 回以下）が指定されており，シナリオのすべてのルートで同じスロットに書き込むステップが 2 つ以上存在しないことが条件となる．

定義 3-1 動作条件定義

動作条件　::＝ 対象＋条件
対象　　::＝ 要素＋範囲
範囲　　::＝ 全体｜特定
要素　　::＝ ステップ｜スロット｜遷移
条件　　::＝ 実行回数｜読込回数｜書込回数｜順序条件

表 3-3 動作条件の作成例

#	要素	範囲	条件
1	スロット	全体	順序条件（書込－＞読込）
2	ステップ	特定（ステップ X）	実行回数（必ず 1 回）
3	スロット	全体	書込回数（1 回以下）

動作条件は以下の 2 種類に分かれる．

・共通動作条件：シナリオのユースケースに依存しない共通的な動作条件であり，シナリオ作成時にデフォルトで設定される．例えば，表 3-3 の#1 の動作条件は，スロットの読込違反を防ぐための動作条件であり，読込違反はユースケースに依存せずどのようなシナリオでも異常であるため，共通動作条件となる．

・カスタム動作条件：シナリオのユースケースに依存する動作条件であり，シナリオの作成者が作成する．表 3-3 の作成例の #2, #3 の動作条件はユースケースに依存する動

作条件（#2 はシナリオの特定のステップを対象としており，#3 の上書きはユースケースによって異常動作とするか変わる）のため，カスタム動作条件となる．

動作条件は，図 3-5 に示すようなシナリオ作成ツールから作成する．共通動作条件は，シナリオの新規作成時に自動で指定されており，シナリオ作成者が必要に応じてカスタム動作条件を専用の入力欄から追加，修正する．

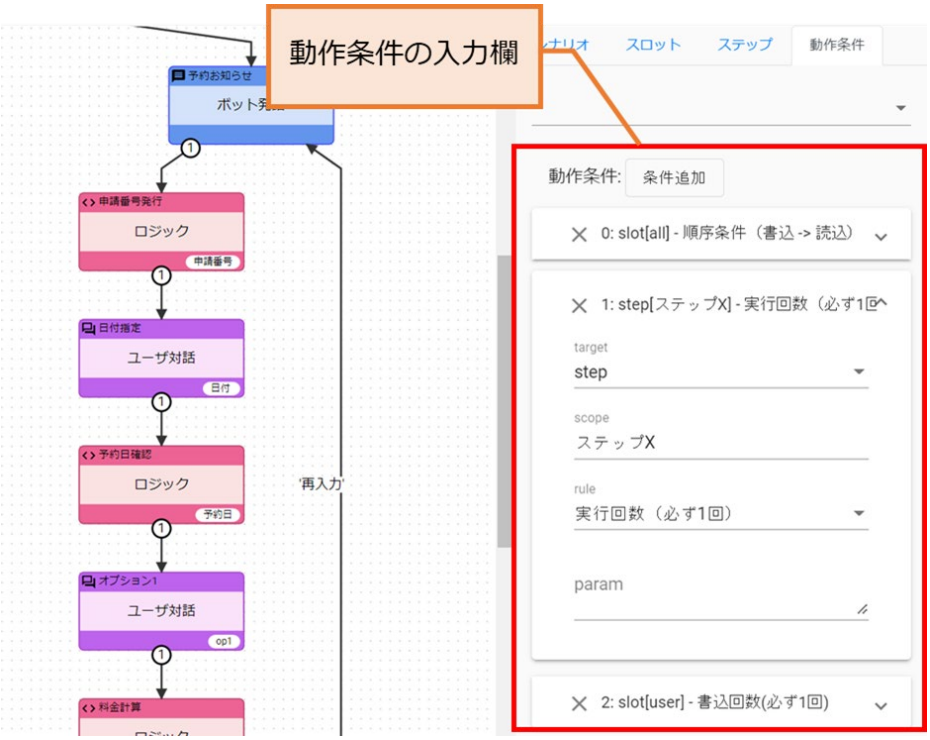

図 3-5 シナリオ作成ツールによる動作条件作成

3.3.3 動作条件によるテストの実行

提案方式では，利用者の発話内容なしにシナリオの構造情報のみからありうるすべてのルートを抽出する．抽出したルート毎に実行されるステップ・遷移の定義情報から動作条件に準拠しているかを機械的にチェックすることでテストを実行する．

動作条件によるテストの実行手順は以下の通りとなる．

(1) シナリオのルートの抽出

作成されたシナリオのステップと遷移の構造（遷移元，遷移先）から，そのシナリオでとりうるすべてのルートを抽出することでテストケースを機械的に作成する．ループ

しているシナリオの場合，無限にルートができるため，事前に指定されたループの打ち切り回数に基づきループを打ち切る．例えば，打ち切り回数として「3」が指定されたとすると，ループ時に同じステップが 3 回まで実行されることとなる．

(2) 動作条件の準拠チェック

シナリオに指定された動作条件に，(1)で抽出した各ルートが準拠しているかを判定し，違反している場合は確認結果として出力する．ルートごとに，実行ステップや遷移の定義情報からステップの実行回数，スロットの読み書き回数，ステップやスロットの順序関係を抽出し，シナリオに紐づけられている動作条件の条件に準拠しているかを判定する．

図 3-1 のシナリオに表 3-3 #1 の動作条件を指定したときを例にテストの実行を具体的に説明する．

まず，シナリオの構造情報から可能性があるルートを網羅的に作成する．図 3-1 のシナリオのステップと遷移の構造から，以下の 4 つのルートが作られる．（ここでループ打ち切り回数は 2 としている）

・ピザ選択ルート

（データ初期化―＞メニュー―＞ピザメニュー―＞ピザオプション選択―＞
注文確認―＞注文ステップ）

・弁当選択ルート

（データ初期化―＞メニュー―＞弁当メニュー―＞注文確認―＞注文ステップ）

・ピザ―キャンセル―弁当ルート

（データ初期化―＞メニュー―＞ピザメニュー―＞ピザオプション選択―＞
注文確認―＞メニュー―＞弁当メニュー―＞注文確認―＞注文ステップ）

・弁当―キャンセル―ピザルート

（データ初期化―＞メニュー―＞弁当メニュー―＞注文確認―＞メニュー―＞
ピザメニュー―＞ピザオプション選択―＞注文確認―＞注文ステップ）

次に作成された各ルートが動作条件に準拠しているかを判定する．各ルートを表 3-3 #1 の動作条件に準拠しているか確認したとき，弁当選択ルート，弁当―キャンセル―ピザルートで「option1」に書き込まれずに「注文確認」で読み込もうとするため，条件

である順序条件（書込－＞読込）に違反しており，異常動作と判定し，作成不良の可能性として出力される．結果として，図 3-6 で示すようにシナリオ作成者に対して，違反した動作条件や発生箇所が提示される．

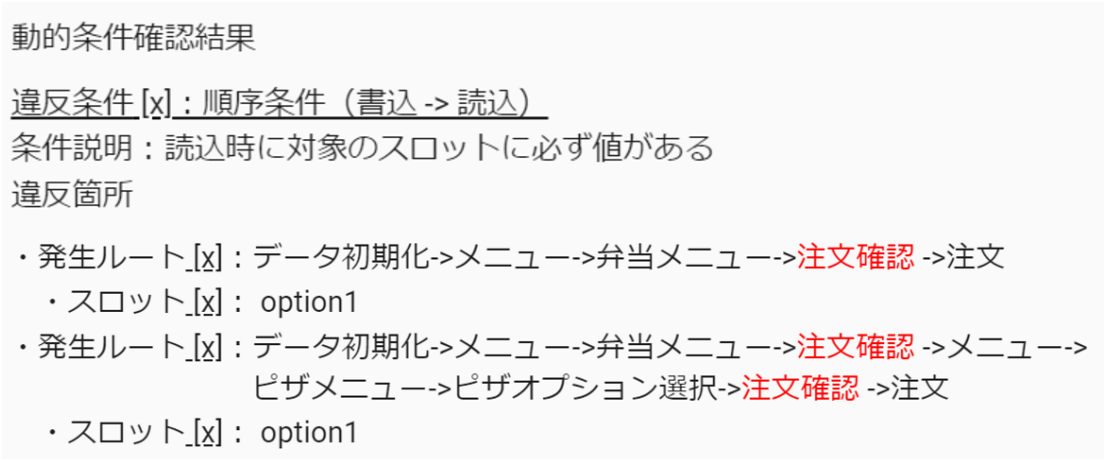

図 3-6 図 3-1 のシナリオに表 3-3 #1 の動作条件指定時のテストの出力例

同じように，図 3-2 のシナリオに表 3-4 の動作条件を指定したときの動作を説明する．まず，以下の 2 つのルートが作られる．（ここでループ打ち切り回数は 2 としている）

・予約確認正常ルート（申請番号発行―＞日時指定―＞予約日確認―＞オプション１―＞料金計算―＞予約確認―＞予約登録）

・予約確認キャンセルルート（申請番号発行―＞日時指定―＞予約日確認―＞オプション１―＞料金計算―＞予約確認―＞申請番号発行―＞日時指定―＞予約日確認―＞オプション１―＞料金計算―＞予約確認―＞予約登録）

その後，指定された表 3-4 の動作条件に準拠しているかを判定すると，予約確認キャンセルルートで表 3-4 の動作条件の対象である「申請番号発行」ステップが 2 回実行されており条件である「実行回数（必ず 1 回）」に違反していることから，異常動作として図 3-7 に示すようにシナリオ作成者に提示される．

表 3-4 図 3-2 のシナリオに対する多重実行を発見する動作条件記述例

#	要素	範囲	条件
1	ステップ	特定（申請番号発行ステップ）	実行回数（必ず１回）

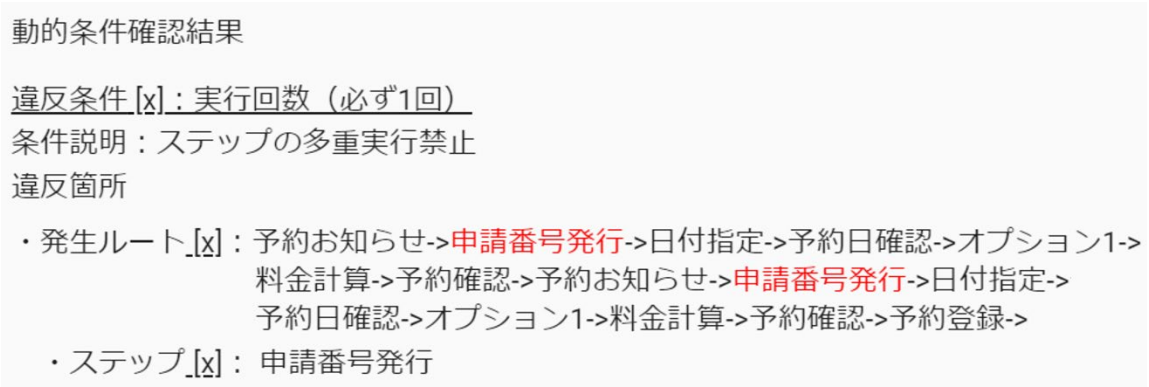

図 3-7 図 3-2 のシナリオに表 3-4 の動作条件指定時のテストの出力例

3.3.4 動作条件の作成支援

シナリオ作成時にデフォルトで作成される共通動作条件に加えて，以下の方法で，動作条件の作成を支援する．

(1) シナリオのテンプレートと動作条件を事前に紐付け

シナリオを作成する際，シナリオをゼロから作るのは時間を要するため，チャットボットシステムから一定のユースケースにあったシナリオのテンプレートが提供されていることが多い．シナリオの作成者は自分が作りたいシナリオにあわせてテンプレートを利用し，一部を拡張する．シナリオテンプレートの例としては，申請登録用テンプレートや，アンケート用テンプレートなどがある．これらのシナリオのテンプレートに，テンプレート用の動作条件を事前に作成し設定しておくことで，ユースケースに依存したカスタム動作条件もシナリオ作成者が独自にシナリオを拡張した箇所以外新たに作成しなくて良くする．例えば，図 3-2 のシナリオが申請登録用テンプレートとすると，申請番号発行の動作条件（表 3-4）はテンプレートの動作条件として事前に作成できる．シナリオの作成者が申請登録用テンプレートを利用する場合，申請番号発行ステップに対する動作条件の作成は必要なくなる．

(2) 動作条件の無効化

動作条件に準拠しない異常動作がシナリオ作成者に提示された際に，確認結果が誤っている（異常とした動作がシナリオ作成者の意図通りで異常動作ではない）場合がある.動作条件が誤っていることになるが,動作条件すべてが誤っているのではなく，特定の要素だけ例外にすべきこともある．一般に確認結果を見て動作条件の作成画面から対象の動作条件を修正するのは手間を要する．このため，確認結果の画面において対象の動作条件あるいはその一部を無効化することを可能にする．無効化できる対象としては，対象の動作条件全て，対象の要素（範囲が「すべて」の時に無効化すると対象の要素だけ無効化される)，対象のルートのみの 3 種類ある．例えば，図 3-8 に示すように，すべてのスロットは書込回数（1 回）とする動作条件によりテストが実行され，確認結果として異常動作の詳細が提示されたとする．このとき，提示された条件やルート,要素のそれぞれに対して,無効化するボタン([x]ボタン)が表示される．「option2」スロットは複数回書き込んでも問題ない仕様だったとした場合，表示されている「option 2」の無効化（[x]）ボタンを押下することで，「option 2」スロットに対してのみ書込回数（1 回以下）とする動作条件を無効化することができる.

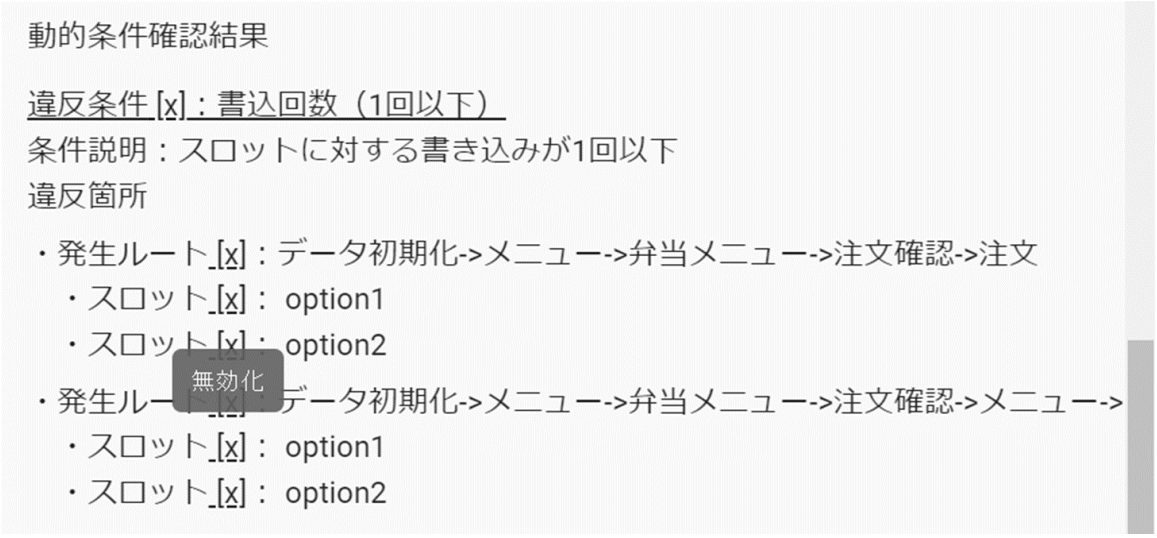

図 3-8 確認結果の提示時の無効化

3.4　評価実験

提案方式による動作条件がシナリオに対して正しく作成できたとして，実際にシナリオを作成した際に発生した作成不良のうち，どの程度提案方式が発見できるかを評価する．作成対象のシナリオは 2 件であり，発生した不良件数は計 21 件となった．このときの人手での作成不良 1 件あたりの発見時間は約 6 分であった．

評価に使用したシナリオのテストで作成した動作条件の数とルートの規模を表 3-5 に示す．共通動作条件は事前に作成されており，すべてのシナリオで共通である．本実験では，シナリオテンプレートを利用しておらずカスタム動作条件はすべてシナリオ作成者が作成した．また，ルート数はシナリオがとりうるすべてのルートの数である．このときのループ打ち切り回数は「2」とした．

表 3-5 シナリオに作成した動作条件の数とルートの規模

シナリオ	共通動作条件	カスタム動作条件	ルート数
A	4	5	80
B	4	3	10

表 3-6 作成不良が提案方式で発見できるかの実験結果

作成不良原因分類	発見可能	発見不可	発見率
スロット指定ミス	12	0	100%
遷移指定ミス	3	1	75%
その他	0	5	0%
計	15	6	71%

実験結果を表 3-6 に示す．結果として，71%の作成不良を提案方式の動作条件により発見できた．特に，スロット指定ミスを原因とする作成不良は 100%発見し，模擬対話方式では，見逃すような複雑なルートで発生するスロットの読込違反やスロットの上書きも発見できた．

この結果から，人手による作成不良の平均発見時間をもとに計算すると，シナリオのテスト時間を約 1 時間半削減し，不良の見逃しの可能性を削減できたと考えられる．また，異常動作が発生するルートや対象の要素もシナリオ作成者に提示されるため作成不

良を容易に修正できることを確認した．

提案方式によるテスト手法で，発見できなかった作成不良は以下のとおりである．

・静的文の間違い（静的指定ミス）

現在の動作条件の条件では発話文などに含まれる誤字などの静的指定ミスが 5 件あり，これらを提案方式で機械的に発見できなかった．これらの作成不良を機械的に発見するには，シナリオ作成者が動作ではなく，厳密な期待値（利用者の発話に対するチャットボットの返答内容）を指定する必要がある．しかし，静的文の間違いは，各ステップを 1 回実行して結果を確認すれば，問題があるか容易に確認できるため構造的な作成不良に比べてシナリオの作成者への負担は少なく，手動で実施したとしても今回のシナリオでは 10 分程度で完了する．

・前に実行したステップに戻る遷移における遷移先設定ミス

利用者が入力した情報を修正する時など利用者が前に実行したステップに遷移するが，この際の遷移先指定ミスを 1 件発見できなかった．具体的には，前に実行したステップに戻る遷移において，利用案内文を発話する発話ステップを遷移先に指定する必要があるのに，異なる（その一つ後に実行する）ステップを遷移先に誤って指定していた．これにより，再入力時に実行するはずのステップが未実行になり，利用者に発話されるはずの利用案内文が表示されない異常が起きた．提案方式の動作条件として，「要素：遷移，範囲：特定（前に実行したステップへ戻る遷移を指定），条件：順序条件（対象要素－＞再入力時の実行ステップを指定）」を追加することで，この作成不良，すなわち，遷移先指定ミスを検出できる．ただし，この動作条件は，チャットボットやプログラミングの知識が浅い業務担当者が作成するには無理がある．テンプレートに組み込む，あるいは，記述の仕方を例とともに与えるなど条件作成を支援することで，動作条件を追加しやすくすることができると考えている．また，シナリオの構造からシナリオ内の利用者からの情報収集のための一連の流れ（ステップ群）を抽出し，戻る遷移の遷移先が適切に指定されている（情報収集のステップ群の先頭に戻っている）かを機械的にチェックする方法なども検討できる．

提案方式における動作条件の作成は，共通動作条件やテンプレートを利用すれば事前に指定されており拡張した部分のみを追加すればよいため，1 シナリオあたり 10 分程度の作業で完了する．また，動作条件による機械的なテスト時間はシナリオの複雑さに

もよるが，実際に運用されるチャットボットのシナリオの規模では，5〜10 秒程度である．

上記から，提案方式によるテストに加えて，手動で各ステップを 1 回実行して結果を確認することで，20 分程度の時間で 9 割以上（提案方式による構造的な作成不良 15 個，手動による静的指定ミス 5 個の計 20 個）の作成不良を発見できる．本実験で見逃した残り 1 割の作成不良は，シナリオ作成者による動作条件の追加をしやすくすれば発見できる可能性がある．また，手動によるテストと提案方式の組み合わせにより，チャットボットの動作が停止するような重大な異常動作を起こす作成不良は発見できていることが想定される．このため，チャットボットの提供スピードを優先し，必要な最低限の品質は確保できているとして，運用を開始後，作成不良が顕在化した際に対応することも検討できる．従来の自動テスト方式と比較すると，従来の自動テスト方式は，想定発話分と返答結果の指定に実験対象のシナリオの規模で，8 時間程度必要な想定であり，20 分程度で完了する提案方式を活用したテスト方法と比べてシナリオ作成者への負荷が高い．従来の自動テスト方式では，作成不良はすべて発見可能だが，チャットボットでは，提供スピードや開発の簡単さが優先されることから，提案方式によるテスト方法の有効性は高いといえる．

3.5　結言

シナリオの作成者が対話時の仕様の動作を簡単にシナリオに記述できるようにすることで，シナリオの作成不良を機械的にテストする方式を提案した．具体的には，シナリオのどこがどのような条件のときにシナリオとして正しいかを指定するシナリオ向きの動作条件定義方法と，シナリオのルートをシナリオの構造のみから抽出し，指定した動作条件に準拠しているかを確認するテスト方法を提案した．結果として，提案方式で，シナリオの作成で発生した 7 割の作成不良を発見した．さらに提案方式に加えて，手動での簡単なテストを組み合わせることで，9 割以上の作成不良を発見できた．これにより，テストの時間や作成不良の見逃しを削減することができ，動作条件の記述時間も小さいことが確認できた．また，厳密な想定発話文と期待値を事前に指定する従来の自動テスト方式と比較した場合，精度は劣るが，テストに必要な時間を大幅に削減でき，チャットボットでは提供スピードや開発の簡単さが優先されることから，提案方式によ

るテスト方法の有効性を確認できた．

今後，シナリオに動作条件の作成を容易化するための，さらなる支援が必要である．提案方式では，共通動作条件やシナリオテンプレートを利用することによりシナリオの作成者がなるべく動作条件を作成する必要がないようにしているが，シナリオを拡張した場合やテンプレートを利用しない場合には，シナリオ作成者がカスタム動作条件の設定を誤って指定すること，デフォルトのまま修正を忘れることが懸念される．正しく動作条件を作成する支援として，シナリオ内の構成要素で動作条件が設定されていない要素を可視化したり，シナリオに指定された動作条件間で整合性が取れているかなどを検査したりすることが考えられる．また，シナリオの構造により自動で動作条件を追加することで動作条件の作成の自動化ができる．例えば，ステップの追加時にそのステップに付与する動作条件を自動で作成することが考えられる．さらに，今回提案手法で発見できなかった戻る遷移時の遷移先指定ミスのチェックのために動作条件の作成を支援することや，静的指定ミスの作成不良を機械的に発見するための手法が求められる．静的な発話文の誤りは，チャットボットの発話文に対して意味解析を利用することで，文法として正しいか，意味を捉えられるかなどの観点から作成不良を検出できる可能性がある．

第4章
インシデント対応迅速化のためのアラート調査支援システム

4.1 緒言

本章では，インシデント対応におけるアラート調査の迅速化に向けて，アラート調査の各種問題を解決するための支援システムを提案する．企業がチャットボットをインターネット上で公開する場合，サイバー攻撃への対応は必須となっている．この対応として，公開しているシステムをセキュリティ機器により監視し，サイバー攻撃の可能性を検出，攻撃であった場合に防御するインシデント対応が重要となる．インシデント対応では，インシデントの疑いが検知された際に優先度の決定や，対応が必要かを判断するトリアージが重要となる．トリアージではファイアウォールや侵入検知システムといったセキュリティ機器が発生させるアラートを調査する．これらのセキュリティ機器は，ネットワークトラフィックを監視し，異常や攻撃の可能性が高いイベントを検知した際にアラートを発生させる．このアラートは多量かつその大部分が問題ないことが分かっている．しかしアラート調査では，アラートの形式がセキュリティ機器により異なることや，調査のためのコマンドの作成，発生アラートの流れを把握するための整形などに，時間を要し迅速化の妨げとなっている．

先行研究として，これらの問題に対し，特定の攻撃調査を対象とした，最適な集計，可視化方式やアラートの変換方式などが提案されている．しかし，アラート調査で実施する様々な調査を支援する可視化方式や共通的なアラート形式を実現することが出来ておらず，インシデント対応を実運用するうえでの迅速化という観点からは不十分である．

そこで，本章では，アラート調査のための支援システムを提案する．このシステムでは，異なるアラート形式を，アラート調査で実施する様々な調査で必要となる項目を持つ共通的な形式に変換する．アラートを攻撃元とアラート間の時間間隔により，2 段階に構造化させることで局所的な発生と継続的な発生を分かりやすく可視化し，分析担当

者が簡単にはわからないような標的型攻撃などの巧妙な攻撃を素早く把握できるようにする．また，アラート調査における調査観点に適した可視化ができるように，可視化に必要な標準的な集計処理をアラート発生時に事前処理する．特定の攻撃が流行した際など必要があればインシデント対応を担当する分析担当者が簡単なフォームで条件を指定し，その条件に従った集計処理を実行できる．

以下，第 4.2 節で，アラート調査の手順や問題点を説明し，第 4.3 節で，提案するシステムを説明し，第 4.4 節で，実際のインシデント対応を例に，提案方式の評価を行う．

4.2　インシデント対応におけるアラート調査

4.2.1 SOC/CSIRT

インシデント対応とは，インシデントを検知した際に優先度の判断・事象分析・対応計画や障害復旧を行うことである．インシデント対応は SOC (Security Operation Center)や CSIRT (Computer Security Incident Response Team)と呼ばれる組織が担当する．狭義には SOC は，ファイアウォールや侵入検知システム(IDS：Intrusion Detection System)[90]などのセキュリティ機器を監視・運用し，インシデントを検知する．これに対して CSIRT は検知されたインシデントに対し適切な対処を実施する．しかし，昨今は一つの組織で双方を実施することが多く，その境界線は曖昧となる[91]．このため，本章は統一的に SOC とする．

また，SOC の形態は大きく二種類ある．

(1)　社内 SOC

社内 SOC は，監視対象の企業内に設置され自社のセキュリティを監視する．後述のアウトソース型 SOC に比べて規模が小さく，監視するセキュリティ機器も限られる．

(2)　アウトソース型 SOC

アウトソース型 SOC は，IT 企業などから SOC サービスやセキュリティデバイス監視サービスとして提供されており，契約した他企業のセキュリティを監視する．アウトソース型 SOC は，多種多様な企業のセキュリティを監視するため，規模が大きく監視対象のセキュリティ機器も多様である．アウトソース型 SOC を活用した場合，その利用企業は比較的小規模なセキュリティに対応する組織を持ち，自組織で実施すべき領域や自組織を中心に連携すべき領域のみに対応し，専門的な知識が必要となる領域を外部

委託する．これは「ミニマムインソース」「ハイブリッド」型と呼ばれる組織形態であり，専門のセキュリティ部門を持たずIT部門がセキュリティも担当するユーザ企業では最も標準的である．本章では主にアウトソース型SOCを利用したこれら二つの形態を対象として説明する．

4.2.2 インシデント対応フロー

インシデント対応の標準的な流れは図 4-1のとおりである[92].

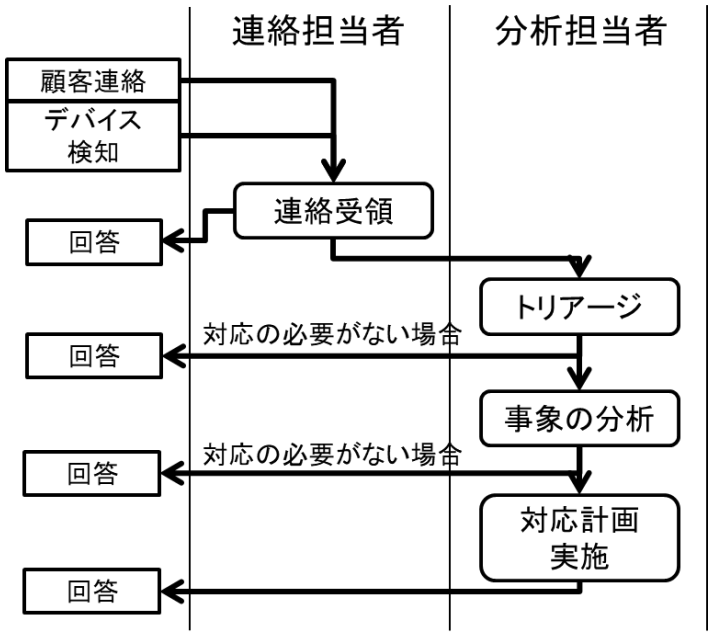

図 4-1 インシデント対応フロー

インシデント対応は，セキュリティ機器など予め用意したシステムによって異常が発見される場合や，監視対象の内部関係者からの異常の連絡，専門組織などの外部からの連絡により開始される．

(1) 連絡受領

検知や連絡を受けた際に，連絡担当者がその情報を受領する．事前に用意された手順書などの判断基準により，その情報が分析担当者に転送するべきインシデントであるかを確認する．連絡された情報が，誤情報や，サーバーのメンテナンスなどの予期された障害など，分析担当者が対応する必要がない場合もある．これらの場合は，分析担当

者には転送せずに, 検知元や連絡元に回答が必要な場合は, その旨を回答し収束させる. もしインシデントの可能性がある場合は，起票し分析担当者に転送する.

(2)　トリアージ

トリアージは，分析担当者が，起票されたインシデントの優先度を判断し，対応が必要か否かを決定する. 誤検知などインシデントではなかった場合や優先度が一定以下の場合などは対応の必要がないと判断する.

この優先度の決定のために，起票情報だけでは足りない場合，追加情報の調査が必要となる．追加情報の調査は下記の順となる.

i.　対応表・過去事例調査

事前に決められた対応表や過去に対応した同様の事例を調査する. 同様事例が存在する場合，同じように対応することが多い.

ii.　セキュリティ機器のアラート調査

アラートは何らかのサイバー攻撃などの疑いが発生した際にセキュリティ機器によって生成され，具体的な攻撃情報などが含まれている．詳細を 4.2.3 節に記述する.

iii.　ネットワークトラフィックのログ調査

アラート調査で疑いが残る場合, プロキシなどのネットワークトラフィックを調査する.

(3)　事象分析

セキュリティ侵害が起きたことを前提に被害の把握や侵害範囲を切り分ける. 侵害の想定度合いや，SOC の形態によっては，マルウェアの感染が疑われる PC（Personal Computer）の詳細な解析やフォレンジック作業を行う．この作業では実環境を模擬した調査環境を構築し，感染したマルウェアを実行し侵害内容を分析する．さらにネットワークトラフィックや PC のイベントを一つずつ追跡して感染の広がりを把握することで，侵害の全容を明らかにする.

(4)　対応計画/実施

事象分析の結果を基に，障害の復旧や情報流出への対応，再発防止策などを計画し，実施する.

4.2.3 アラート調査

アラート調査では，セキュリティ機器が発生させるアラートを調査する．アラートは統合脅威管理(UTM：Unified Threat Management)[93]，IDS やファイアウォールなどのセキュリティ機器が，ネットワークトラフィックを監視し，異常や攻撃の可能性が高いイベントを検知した際に発生させる．例えば，図 4-2，図 4-3 はセキュリティ機器が発生させたアラートの一例である．これらのアラートは，少しでも攻撃の可能性があると発生するが，実際の攻撃につながるのは一部である[94]．このため，アラートが発生した際には攻撃の可能性を調査する必要がある．アラート調査はイベント調査と統計調査に分かれる．

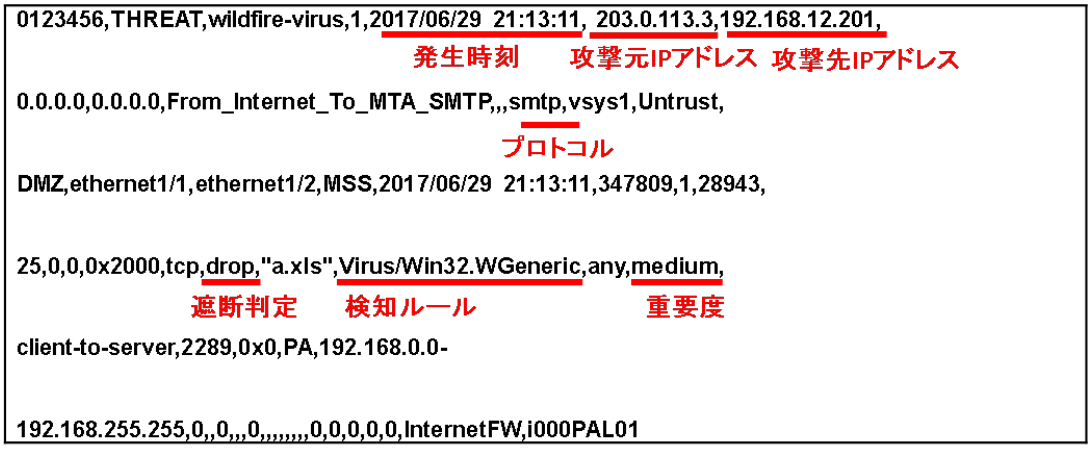

図 4-2 UTM(Palo Alto Networks 社)のアラートログ

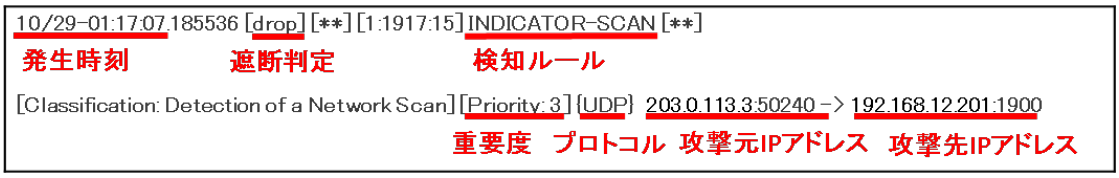

図 4-3 UTM(Snort IDS)のアラートログ

4.2.3.1 イベント調査

イベント調査では起票情報を基に発生したアラートを調査する．主な調査観点は下記のとおりである．

・日付

・攻撃元 IP (Internet Protocol)アドレス

・検知ルール

・攻撃先 IP アドレス

・遮断判定

・重要度

起票に関連したアラートを調査し不審な点がないか確認する. いつ発生したかではなく, 上記の調査観点に加えて局所的もしくは継続的なアラートの流れからの多角的な調査が重要となる.

i. 局所的発生

時間的な間隔がなく連続して発生しているアラートを調査する. 特に複数の検知ルールによってアラートが発生している場合は, 意図を持った攻撃の可能性が高いとされる. 例えば, 公開ウェブページに対する攻撃などがある. この場合, SQL インジェクションやディレクトリトラバーサル, 不正ログインの試み, ポートスキャンなど複数ルールによるアラートが短期間に連続して検知される.

ii. 継続的発生

継続的に発生しているアラートを調査する. 長期間にわたり継続的に発生している場合は, 端末がマルウェアに感染し, 何らかの情報を外部に発信している可能性などが挙げられる. また, 標的型攻撃などは IDS などの検知を逃れるため, 長期間にわたり攻撃する. これらのインシデントの発見のため, 長期間にわたり同一の攻撃元や攻撃先に関連するアラートが発生していないかを調査する.

4.2.3.2 統計調査

統計調査では, アラート数などの時間的変化を調査し, 長期的な傾向から攻撃の兆候が発生していないか確認する. 統計調査は一件の攻撃に対応する面よりも, 社会的なサイバー攻撃の発生の予兆などを把握し, セキュリティ体制を高度化する事前対策の面が強い.

アラート調査の標準的な調査工程は, 下記のとおりである.

(1). 起票情報からアラートを取得し, 関係するセキュリティ機器への接続に必要なログイン情報を取得する.

(2). 対象のセキュリティ機器やアラートが蓄積されているサーバーにアクセスする．

アラートの蓄積方式は，セキュリティ機器の機種や，設置されているネットワークによって異なる．そのため，調査対象の機種やネットワークに応じた手順書を確認し，アラートが蓄積されている機器・サーバーなどにアクセスする必要がある．

(3). 調査観点に応じたアラートを抽出するコマンドを作成する．

調査手順書を確認し，調査する観点・機種に応じてアラートを抽出するコマンドを作成する．アラートは，機種によってフォーマットが異なるため，作成するコマンドの対象ファイルのパスやパラメータ名を変更する必要がある．例えば，図 4-4 のコマンドは，Palo Alto Networks 社製 UTM の特定期間のアラートが蓄積されているファイルから，特定の攻撃元 IP アドレスかつ検知ルールを条件に抽出するものである．これに対して, Snort IDS[95]から同様の条件で抽出するコマンドは図 4-5 に示すものとなり，対象とするファイルやパラメータが異なる．ファイルのパスは，機器の設定で統一することもできる．しかし，分析担当者が誤って違う機種用のコマンドを実行した場合，アラート形式が異なり, 何もアラートを抽出できず想定した検知ルールのログは発生していないと誤解したり，コマンドの意図と異なるアラートを抽出した結果，攻撃を見逃したりする可能性がある．そのため，運用上の誤操作防止の観点から，アラート形式が異なる機種のアラートを蓄積するファイルのパスを変えていることが多い．

```
zcat /var/log/palo/logs/vulnerability/vulnerability.log-201810*.gz |
    grep -i '{攻撃元IPアドレス}' | awk -F',' '{$30=="{検知ルール}"){print $0}' > temp_alert.txt
```

図 4-4 特定の攻撃元 IP アドレスかつ，特定の検知ルールのアラートを抽出するコマンド（Palo Alto Networks 社 UTM）

```
zcat /var/log/snort/logs/alert.log-201810*.gz |
    grep -i '{攻撃元IPアドレス}' | awk '{$5=="{検知ルール}"){print $0}' > temp_alert.txt
```

図 4-5 特定の攻撃元 IP アドレスかつ，特定の検知ルールのアラートを抽出するコマンド（Snort IDS）

また，調査観点毎に，コマンドを作成する必要がある．例えば，調査観点の一つとして，セキュリティ機器が遮断したアラート件数の時間的な変化を調査する場合がある．このコマンドは図 4-6 であり，図 4-4 のコマンドと異なっている．このように，調査観点により，コマンドの種類や数，パラメータを変更する必要がある．

```
zcat /var/log/palo/logs/vulnerability/vulnerability.log-201806*.gz
        | grep drop | awk -F',' '{print $5}' | cut -b 1-10 | sort | uniq -c
```

図 4-6 特定の攻撃元 IP アドレスかつ，遮断したアラートの日毎件数を出力するコマンド(Palo Alto Networks 社 UTM)

(4). (3)で作成したコマンドを，アラートファイルに実行する．

(5). 結果を表計算ツールなどに記録する．

調査観点が複数の場合や，複数の時間帯を調べる必要がある場合，(3)から繰り返す必要がある．対象の機器が複数の場合は，接続機器を変更して(2)から繰り返す

(6). 記録した結果を人が判断しやすいように整形する．

局所性と継続性を見るために，表計算ツールなどで，加工する．図 4-7 は，特定の攻撃元 IP アドレスかつ，特定の検知ルールのアラートを抽出した結果（図 4-4 のコマンドの実行結果）を記録し，アラート間の時間間隔を表示したものである．1 時間以上の間隔で，2 回の局所的なアラートが発生していることがわかる．

(7). 結果から，重要度や対応を判断する．判断に必要な情報が不足していた場合，(2)に戻り，追加で調査する．

発生日時	間隔	UTM名	アラート
2017/5/9 18:16		Paloal	0123456,THREAT,wildfire-virus,1,2017/05/
2017/5/9 18:16	0:00:22	Paloal	0123456,THREAT,wildfire-virus,1,2017/05/
2017/5/9 20:21	2:04:38	Paloal	0123456,THREAT,wildfire-virus,1,2017/05/
2017/5/9 20:23	0:02:13	Paloal	0123456,THREAT,wildfire-virus,1,2017/05/
2017/5/9 20:23	0:00:10	Paloal	0123456,THREAT,wildfire-virus,1,2017/05/
2017/5/9 20:24	0:00:39	Paloal	0123456,THREAT,wildfire-virus,1,2017/05/
2017/5/9 20:55	0:31:21	Paloal	0123456,THREAT,wildfire-virus,1,2017/05/
2017/5/9 22:25	1:29:37	Paloal	0123456,THREAT,wildfire-virus,1,2017/05/
2017/5/9 22:26	0:01:11	Paloal	0123456,THREAT,wildfire-virus,1,2017/05/
2017/5/9 22:26	0:00:00	Paloal	0123456,THREAT,wildfire-virus,1,2017/05/
2017/5/10 0:32	2:05:49	Paloal	0123456,THREAT,wildfire-virus,1,2017/05/
2017/5/10 1:36	1:04:00	Paloal	0123456,THREAT,wildfire-virus,1,2017/05/

図 4-7 表計算ツールによるアラートの整形

4.2.4 アラート調査の問題点

アラート調査は短時間で完了する必要があるが，下記の課題により時間を要し問題となる．

(1) セキュリティ機器によりアラート形式が異なる．

種類やメーカーが違うセキュリティ機器のアラートは構文や項目名の形式が異なる．特にアウトソース型 SOC では多種のセキュリティ機器を監視している．その結果，コマンド作成する際の対象となるファイルのパスや，パラメータ名が変わる．

(2) 調査観点ごとにコマンドを作成する必要がある．

アラートの調査観点ごとに対応したコマンドを作成しなければならない．例えば，一定期間内に発生したアラートの攻撃元が過去関係したアラートを調査する場合，攻撃元 IP アドレスをリストアップし，攻撃元 IP アドレスの数だけ，コマンドのパラメータを変え，実行する必要がある．

(3) 局所的・継続的なアラート発生の調査に整形を必要とする．

アラートの局所性や継続性を確認するために，一定期間のアラート群を，人が解釈しやすいように表計算ツールなどで整形する必要がある．

4.3　多角的観点からの調査を支援するアラート調査支援システム

4.3.1　システム概要

本章ではアラートをさまざまな定型パターンで分析，可視化し，分析担当者によるアラート調査に必要となる様々な調査観点による分析に加えて局所性，継続性の分析を含めた多角的な観点からの調査を支援するシステムを提案する．本システムの構成を図 4-8 に示す．本システムは，デバイスが発生させる形式が異なるアラートを統一的な形式に変換し，攻撃元 IP アドレスとアラート間の時間間隔により構造化することでアラートの局所性・継続性をわかりやすく可視化できるようにする．また集計処理を二つに分け，標準的な集計はアラート発生時に事前に処理する．他の集計が追加で必要となった場合，専用のフォームから分析担当者はリクエストできる．分析担当者は集計結果をパーツとして画面上のダッシュボードに好きなように配置でき，担当範囲により必要な可視化結果のみ表示できる(図 4-9)．提案システムでのダッシュボードや集計基盤は Piwik[96]を利用し，アラート調査向けに改造している．分析担当者による多角的な調査観点によるアラート調査に必要な情報を集計し，それぞれパーツとして可視化できるようにすることで，アラート調査を迅速化する．

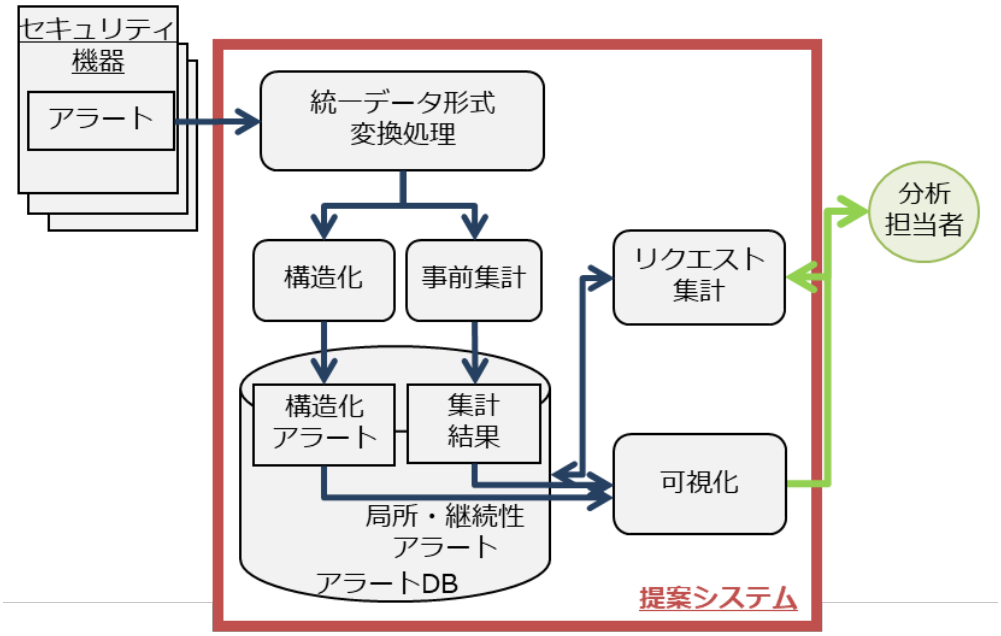

図 4-8 システム構成図

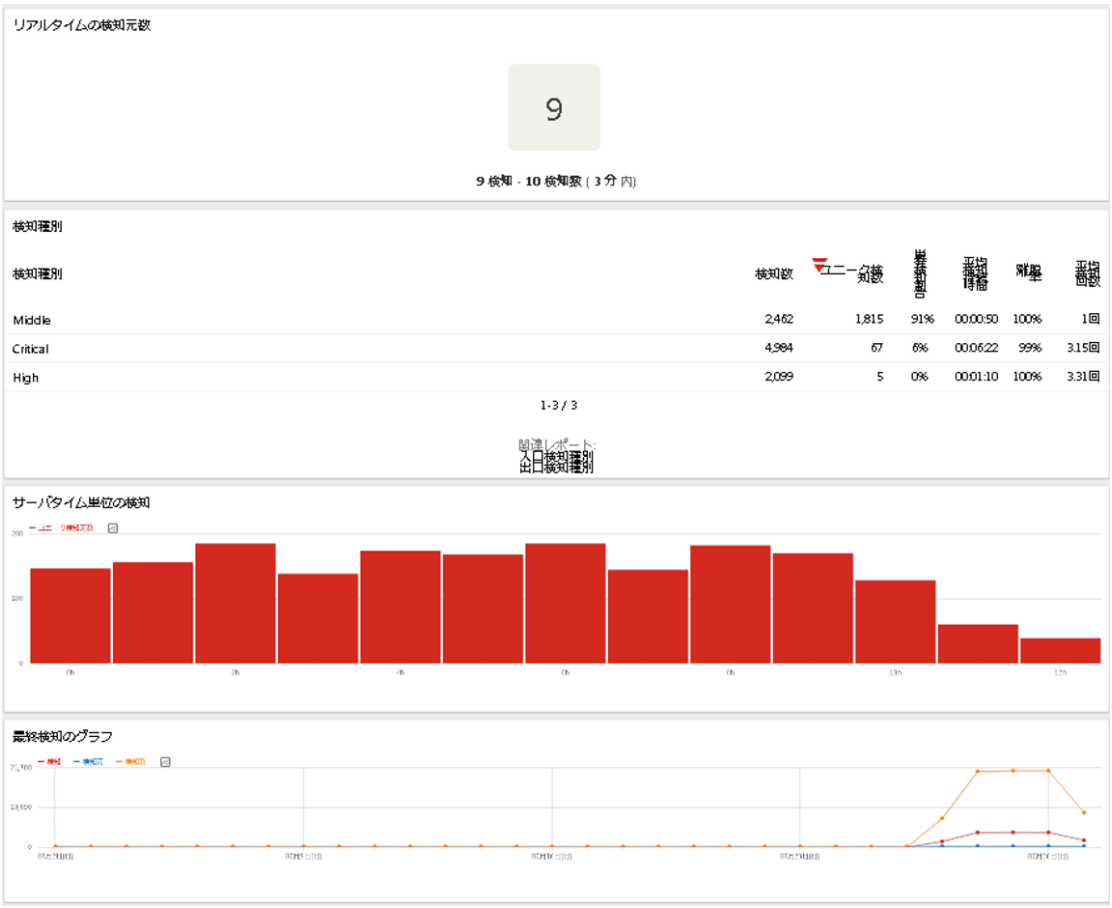

図 4-9 提案システムの可視化結果

4.3.2 統一的データ形式への変換方式

形式が異なるアラートに対応するため統一的データ形式を定義する．これにより，多種の機種のアラートを同一の形式に変換し，調査観点の集計・可視化に必要な処理を統一する．アラートを統一形式に変換し，DB に蓄積することで，機種によりコマンドの対象ファイルのパスやパラメータの差異を気にせず，分析担当者は，調査対象の統一的な操作で，全ての機種のアラートを調査することができる．

IDMEFのフォーマットを参考に,アラート調査に最低限必要な項目を共通部として,それ以外の追加項目を,カスタム部として表 4-1 に示す統一的データ形式に変換する．共通部は，セキュリティ機器のアラートの標準的な項目を持つ．カスタム部には共通化できない項目を格納する.このためカスタム部は,任意の数を任意のキーで指定できる.

アラートの統一的データ形式への変換は，fluentd[74]を基にしており，定義ファイルで設定が可能である．定義ファイルは，アラート変換のみを対象として，fluentd と比べて簡略化しており，データ変換への知識を持たない分析担当者でも定義が可能である．

表 4-1　統一的データ形式

key		項目名
共通部	logtype	アラートの種類
	time	発生時刻
	devid	機器識別番号
	attackerip	攻撃元 IP アドレス
	attackerport	攻撃元ポート
	victimip	攻撃先 IP アドレス
	victimport	攻撃先ポート
	severity	重要度
	attack	検知ルール
	protocol	通信プロトコル
	action	遮断判定
カスタム部		任意（スキーマレス）

定義ファイルの更新は，それほど高頻度では必要がないが，新しい種類のセキュリティ機器の追加や，セキュリティ機器の更新によりデータ形式が変更されたときに必要となる．そのため，分析担当者が，定義ファイルを作成・変更できることは，重要である．

定義は大きくアラート入力定義とアラートマッピング定義の二つに分かれる．アラート入力定義は，図 4-10 のとおり対象となるアラートの入力方法を定義する．基本的には入力のプロトコル(file,syslog,http)と各プロトコルに必要な情報を記述する．入力アラートの形式が file 監視の場合は，type に tail と指定することで，path のファイルを監視し，追記されたときにそのデータを処理する．http の場合は，受信ポートを指定することで，[https://domain:port/tag]の形式でアラートを受信する．ここでの tag とは，マッピング定義を指定するための識別子である．

アラートマッピング定義は図 4-11 となり，key と，アラートから項目を抜き出すた

めの形式を指定する．${} 内は，アラートから項目を抜き出すためのマッピング指定を行う．マッピング指定は対象アラートのフォーマットにより異なる．例えば csv の場合は，format を csv とし，csv_param[列番(0…n)] とする．区切り文字がない形式の場合，format を none とし，アラートログに対する正規表現によって指定する．例えば「…Action=”{遮断判定}”…」の場合，(/^.*?Action="([^"]*)".*$/)となる．この正規表現の書式は，ruby 言語に準拠する[97]．カスタム部[custom_name1~n]は，任意の key 名で，セキュリティ機器が独自に持つパラメータを設定することができる．また，セキュリティ機器により，分析に必要な項目の一部が，別のファイルにリスト化されている．例えば，検知されたルールが，なぜ発生したかの説明項目などである．これに対応するため，別ファイルとアラートの項目を対応付け，別ファイルのデータを挿入することができる．

```
# アラート入力定義
## ファイル or syslogの場合
<source>
  type {tail|syslog}
  tag {タグ名}
  path /var/log/palo6.log
</source>

## HTTPの場合
<source>
  type http
  port 8888
</source>
```

図 4-10　アラート入力定義の形式

```
# アラートマッピング定義
<match {タグ名}>
 ## 共通部
 format ${none|csv|json}
 logtype ${}
 time ${}
 devid ${}
 attackerip ${}
 victimpip ${}
 attackerport ${}
 victimport ${}
 severity ${}
 proto ${}
 attack $}
 action ${}
 ## カスタム部
 custom_name1 ${}
 custom_name2 ${}
 custom_name3 ${}
</match>
```

図 4-11 アラートマッピング定義

4.3.3 テンプレートベースの集計機能

調査のためのコマンドの作成時間を削減するため，図 4-12 に示すテンプレートベースの二つの集計機能を提供する．テンプレートでは画面上の簡単なフォームで条件を入力し，実行タイミングを事前集計機能もしくはリクエスト集計機能に設定することで作成できる．

一つ目の事前集計機能では，アラート調査に必要な標準的な集計機能をテンプレートとして事前に定義し，アラート発生時に集計する．主に統計調査に必要な調査観点別の集計を行う．事前に集計することで，複雑な条件や大規模なデータに対する処理で高速化が望める．なお，登録されていない機能が必要になる場合は，随時テンプレートを追加登録することができる．

二つ目のリクエスト集計機能では，分析担当者は必要になった際に，テンプレートを作成し実行する．主にキーワードベースの抽出や指定の期間での集計に利用される．リ

クエスト集計機能では，一度実行した集計処理はテンプレートとして記録されるため，再度実行する際は条件の再入力が不要になり効率的に集計が可能となる．

図 4-12 テンプレートの作成画面

4.3.4 アラートの構造化

局所的なアラートと継続的なアラートを分かりやすく可視化するために，提案システムではアラートを二段階に構造化する．図 4-13 はアラートの構造化モデルである．同一攻撃元 IP アドレスのアラートで，アラート間の時間間隔が一定以下の場合に一つの「攻撃」として関連付ける．複数の「攻撃」は一定時間ごとに「攻撃元」と関連付けられる．

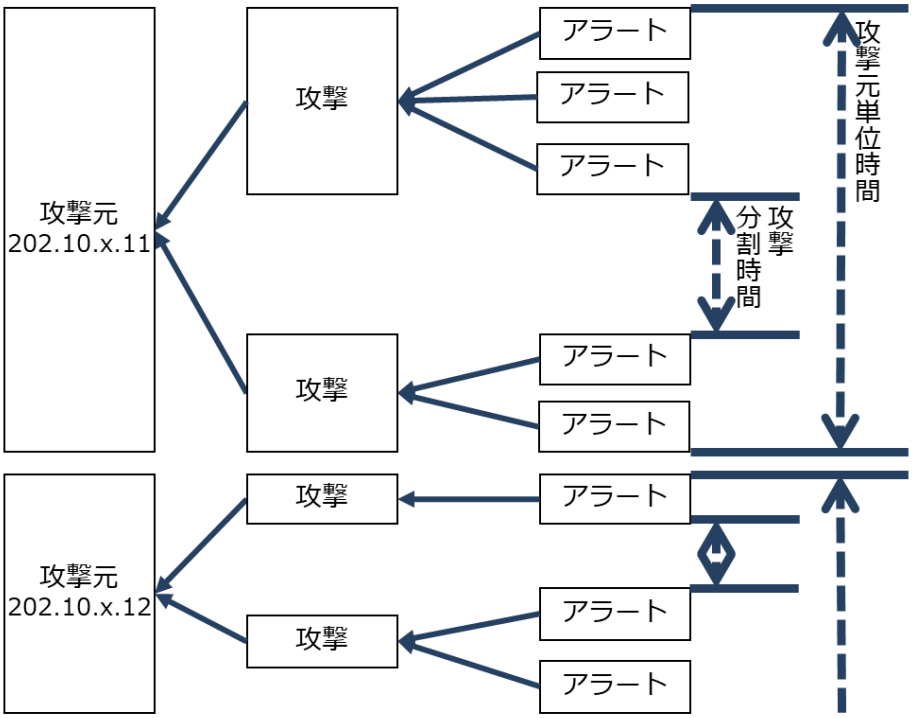

図 4-13 アラートの構造化モデル

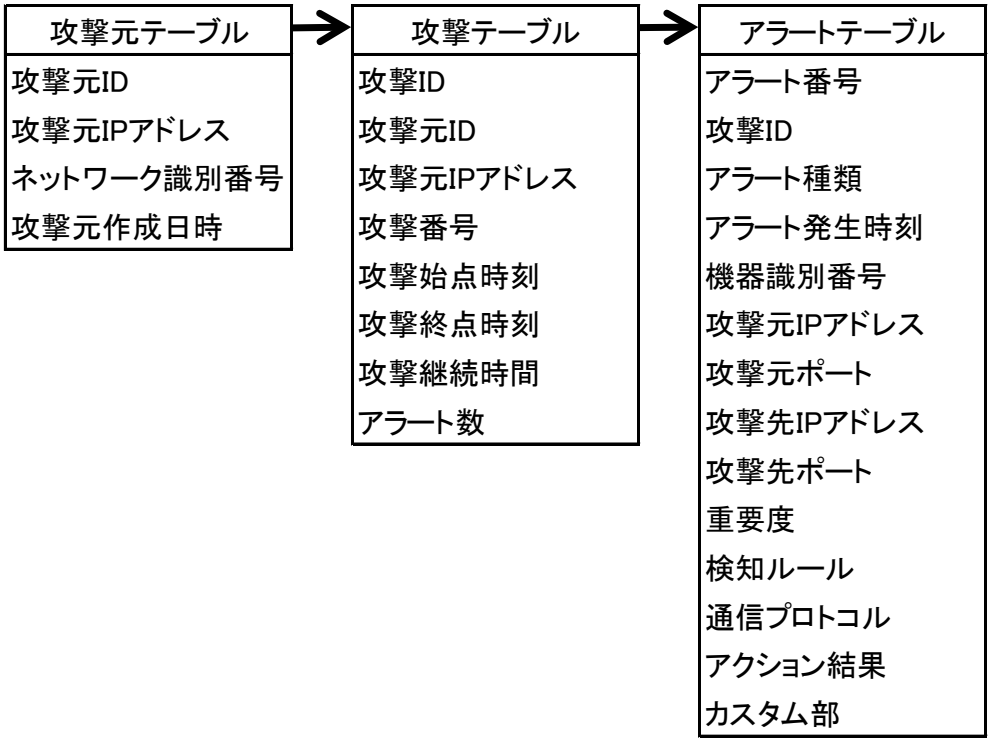

図 4-14 構造化したアラートの論理スキーマ

図 4-14 はアラートを構造化した際の論理スキーマである．

(1)　アラートテーブル

アラートテーブルには，一意な ID を付与されたセキュリティ機器のアラートが統一的データ形式で格納される．

(2)　攻撃テーブル

攻撃テーブルには，ネットワーク識別情報，一意な ID と攻撃回数を表す攻撃番号，攻撃の始点・終点時刻・継続時間，関連付けられているアラートの数を格納する．攻撃テーブルは，アラートテーブルと攻撃 ID によって関連付けが行われる．ここでの攻撃の始点，終点時刻とは，関連付けられているアラートの内，最初に関連付けられたアラートの時刻を始点時刻，最後に関連付けられたアラートの時刻を終点時刻としたものである．これにより一回の攻撃が何時から何時まで続いたのかを容易に把握できる．

(3)　攻撃元テーブル

攻撃元テーブルには，攻撃元 IP アドレスとネットワーク識別番号，攻撃元作成日付を格納する．攻撃元テーブルは攻撃テーブルと攻撃元 ID によって関連付けられる．

図 4-15 は構造化したアラートの可視化画面である．「攻撃」ごとにアラートを分割して可視化しており，一つの「攻撃」内のアラートは局所性があることが分かる．「攻撃元」内の「攻撃」の数によりその攻撃元からの攻撃の継続性が分かる．また，アラートは一覧化され，必要であれば具体的な情報も可視化できる．分析担当者は不審な「攻撃」のアラート 1 件ごとの中身を追うことで，アラートの優先度を判断できる．このように単純にアラートを表示するよりも，局所的なアラートやアラートの流れを分析担当者が確認しやすい．

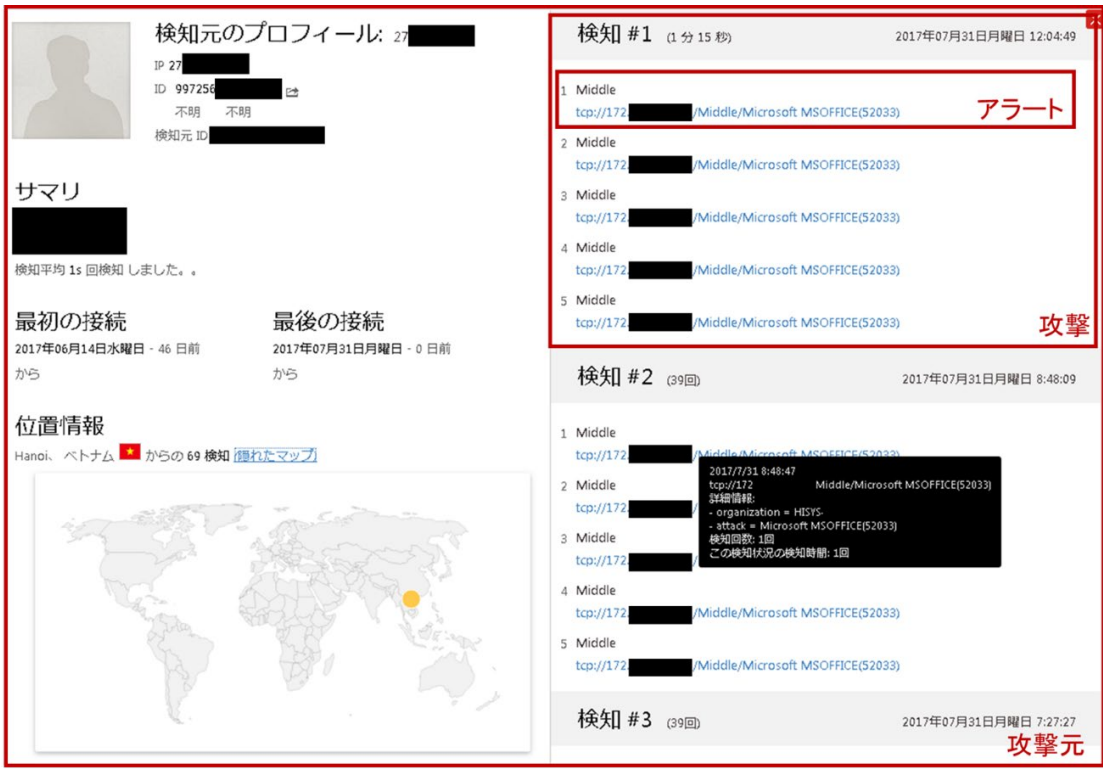

図 4-15 構造化アラートの可視化画面

(攻撃元・攻撃先 IP アドレスなどはマスキング済み)

4.3.5 調査手順の比較

提案システムと既存システムのアラート調査手順を比較したものを図 4-16 に示す. 既存システムと提案システムにおけるアラート調査手順の違いは下記の点となる.

・提案システムでは，事前にセキュリティ機器からアラートを収集しており，既存システムのようにアラート調査の度に，調査対象のセキュリティ機器に接続する必要がなくなる.

・提案システムでは，アラート調査観点に必要となる標準的な集計機能を事前に処理しているため（2）テンプレート選択するだけで必要な情報が入手できる．これに対して既存システムの手順では，(3) 調査用のコマンドを作成し，(4) コマンドを実行することが必要である.

・提案システムでは，アラートを局所，継続性を確認するために，アラート発生時に，自動的に構造化しており，必要な際に確認できる. したがって，既存システムの手順(6)のように，アラートの局所性や継続性を見るために，整形する必要がなくなる.

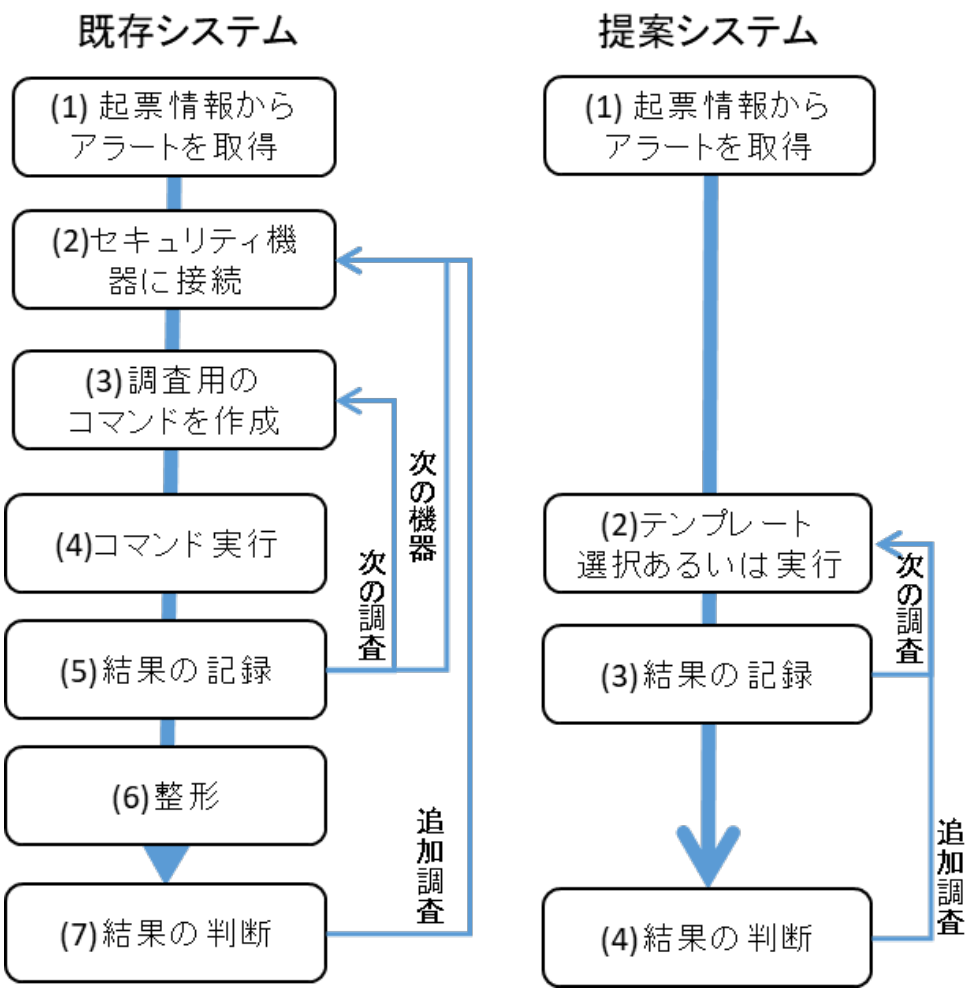

図 4-16 既存システムと提案システムのアラート調査手順

4.4 評価実験

4.4.1 評価環境

評価として提案システムを試行し，既存システムと入力回数・アラート調査に要した時間を比較した．ここでの既存システムとは，セキュリティ機器のアラートをコマンドベースで検索するシステムを指す．

本実験は 4.2.1 節で述べたハイブリッド形態におけるアウトソース型 SOC で実施した．この SOC では既存システムが稼働中であり，並行して提案システムを稼働させた．評価時の対象機器は Palo Alto Networks 社，Check Point 社，Fortinet 社の UTM である． UTM は，IDS やアンチウィルス，ウェブフィルタリングなどの異なる機能を持

った複数のセキュリティツールを一つにまとめたものであり，統合脅威管理とも呼ばれる．このため一つの機器からさまざまな種類のアラートを発生させる．例えば Palo Alto Networks 社の UTM では，インシデント対応において重要となる以下 3 種類のアラートを発生させる[98].

・脅威アラート

・URL フィルタリングアラート

・WildFire(サンドボックスマルウェア分析)アラート

提案システムでは，事前にこれらの機器のインシデント対応に重要となるアラートを取り込めるように，定義ファイルを定義した．

実験対象ネットワークは，4 ネットワークであり，内，3 ネットワークは 1 機種のセキュリティ機器，1 ネットワークは 2 機種のセキュリティ機器が設置され，平均 3 種類のアラートを発生させる．1 ネットワーク当たり 1 ヶ月平均で約 84,000 アラートを発生させる．アラートは，それぞれのネットワーク内の専用サーバーに一度蓄積される．蓄積されたアラートは，4 台の収集サーバーによって，定期的に統一的データ形式へ変換される．変換されたアラートは 1 台の DB・可視化サーバーにより，可視化する．これらのサーバーは，アラート量や，システムへの接続数に応じて，台数を増やし，分散させることができる．

ネットワーク上での実験の結果，提案システムにおいて，一つの「攻撃」に平均 17.4 件のアラートが関連付けられ，「攻撃元」には平均 2.9 件の「攻撃」が関連付けられた．なお，分析担当者の意見のもと，実験での攻撃元分割時間は 1 時間，攻撃元単位時間は 1 週間とした．また，提案システムの初期状態として可視化されるパーツは，分析担当者の要望によりカスタマイズしている．このカスタマイズは提案時のヒアリングにより実施され，その後のチューニングは行っていない．

実験環境上の提案システムにおける処理例を，図 4-17 および図 4-18 に示す．この例では，一つのネットワーク A 上に二つの異なる機器があり，それぞれのアラート[p1~pn,c1~cn] を定義に沿って統一的データ形式に変換している．その統一的データ形式のアラートを元に，テンプレートの実行による集計結果がダッシュボードに可視化される．

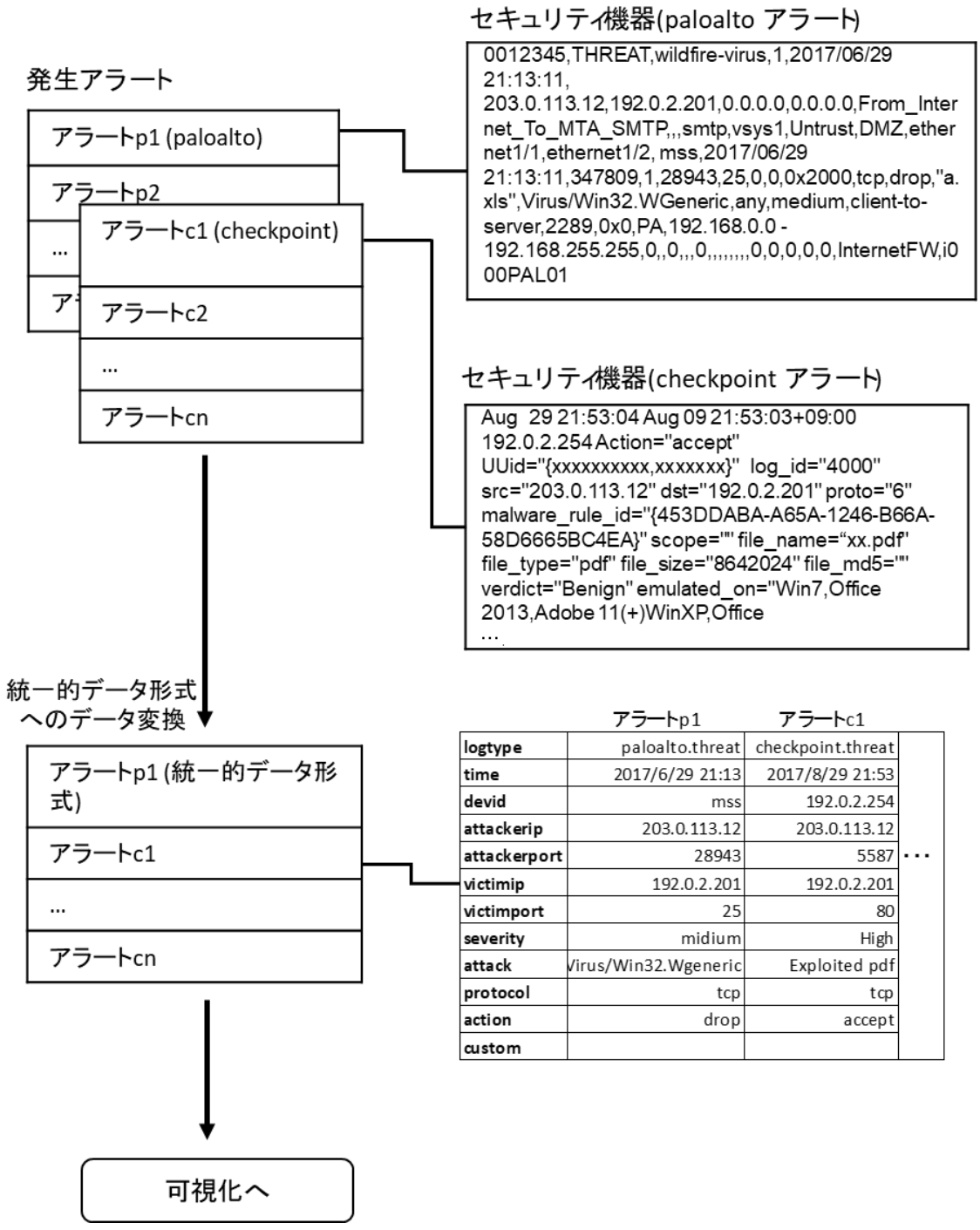

図 4-17 提案システムの動作例：アラートの発生からデータ変換

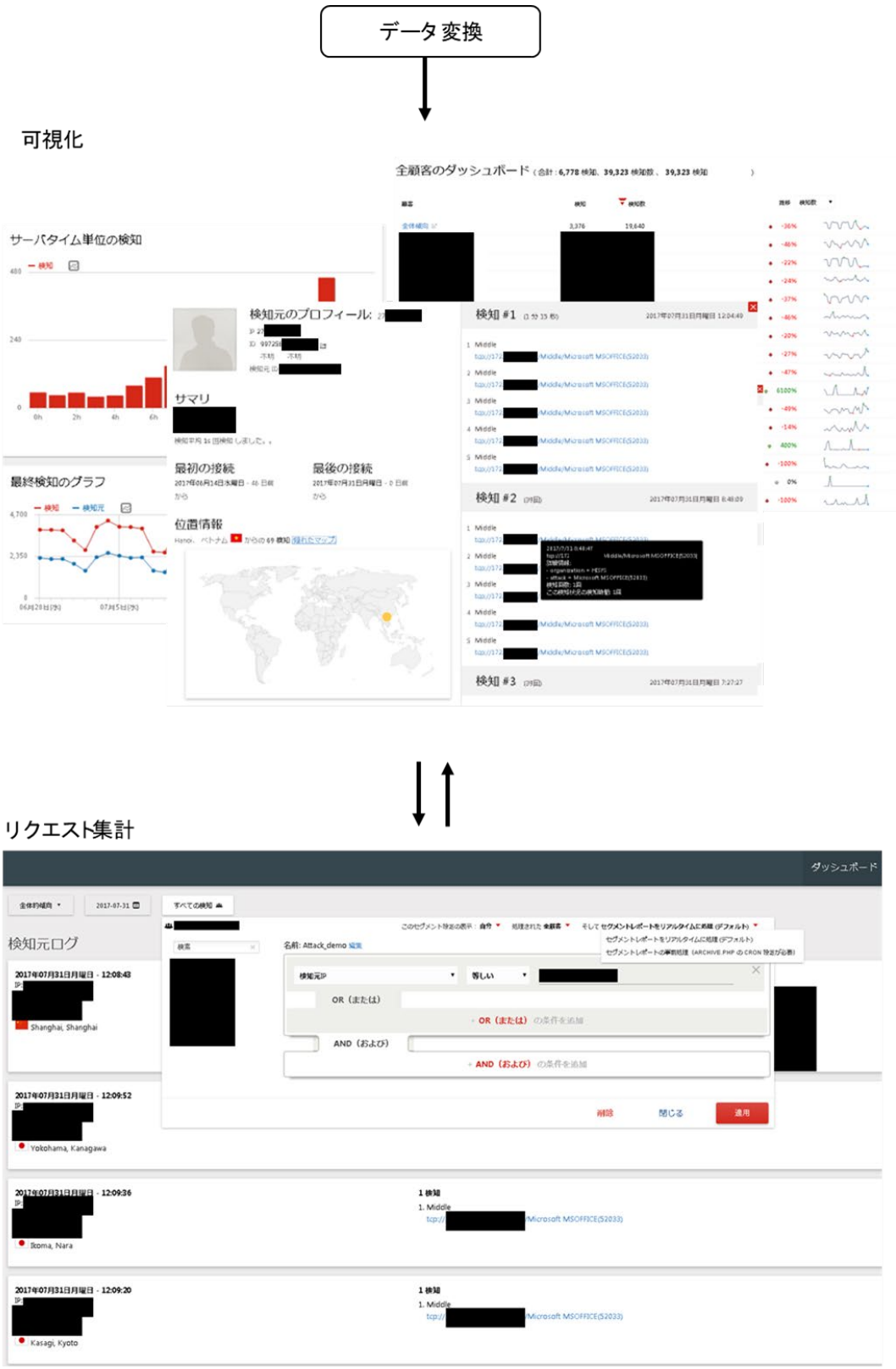

図 4-18 提案システムの動作例：可視化およびリクエスト集計

(攻撃元・攻撃先 IP アドレスなどはマスキング済み)

4.4.2 入力回数の比較

テンプレートを整備することにより，どの程度コマンドが集約され，どの程度入力回数が減るのかをアラート調査の調査観点ごとに把握するため，過去のアラート調査事例 11 件を元に既存システムと提案システムでの入力回数を比較した．SOC における分析担当者による既存システムのコマンド入力ログを元に，提案システムで同様の調査結果を確認できるまでを模擬し，その入力回数を記録した．入力回数は対象機器，システムへのログイン後から記録した．なお，提案システムによる対象事例 11 件のアラート調査では，既存システムと同じ調査結果を分析担当者が作成できることがわかり，提案システムによる発見漏れや判断誤りが起きないことを確認できた．また，統一的データ形式に変換したアラートで，アラート調査で実施する様々な調査に必要となる可視化が可能であることを確認した．

それぞれの入力回数は下記のとおりである．

・既存システム(コマンド)　　：1 回のコマンド作成

・提案システム　　：1 回のテンプレート選択

既存システムのコマンド入力ログでは，ミスタイプによる誤操作は除去している．また，提案システムの入力回数は，既存システムのコマンド入力の結果と同様な結果を，提案システムにおいて表示できるまでの最適なテンプレート選択の回数となる．

表 4-2 アラート調査 1 回あたりの入力回数の比較結果

調査観点	既存システム(コマンド)	提案システム
日付	1.73	0.55
攻撃元 IP アドレス	3.18	0.45
検知ルール	2.82	1.00
攻撃先 IP アドレス	0.36	0.09
遮断判定	2.18	0.73
重要度	1.00	0.36
その他	0.91	0.18
入力回数合計	12.18	3.36

アラート調査事例 11 件における調査 1 回あたりの入力回数の平均を調査観点別に比較したものを表 4-2 に示す．表 4-2 の調査観点は 4.2.3 節で説明したアラート調査内のイベント調査で分析担当者によりアラートが不審でないかを判断するために確認する観点である．各調査観点で平均入力回数が 1 回以下となることが生じる理由は，調査する必要のない観点が存在するためである．例えば，分析担当者が，ある観点で調査した結果，アラートに問題がないと判断した場合，そこで調査を打ち切る．この場合，それ以後の調査観点は省略される．また，本実験ではリクエスト集計は実施されなかった．

既存システムと提案システムを比べると，調査 1 回あたりの平均入力回数が 72.4%削減された．この内，攻撃元 IP アドレスの調査時の入力回数が約 7 分の 1 となり，最も削減された．これは，攻撃元 IP アドレス調査のための調査作業は，既存システムでは平均入力回数が 3.18 回と他の調査観点の平均 1.50 回に比べ多数のコマンドを必要とするが，比較的定型的な作業が多く，提案システムでは，それらをまとめて 1 つのテンプレートとすることで集約できたことが要因である．また，全体的に入力回数が減っているのは，テンプレートの整備により，繰り返しコマンドを作成・実行する必要がなくなったことが要因である．

4.4.3 必要時間の比較

提案システムを実際に試行して既存システムと比較した結果を表 4-3 に示す．分析担当者が提案システムを試用してアラート調査を実施した際の利用ログから，調査に要した時間を求めた．既存システムの場合は，提案システムを試用した分析担当者が既存システムで対応した 19 件の実際のインシデント対応時間を記録した．なお，両者のインシデントの複雑さは同程度である．

表 4-3 調査 1 件あたりの必要時間の比較結果

	件数	調査一件当たりの必要時間[分]			
		平均	最小	最大	中央
既存システム(コマンド)	19	22.3	4.9	58.4	19.6
提案システム	11	8.2	2.0	24.6	7.0

アラート調査一件当たりの平均必要時間を既存システムと比べると提案システムでは 63.1%削減できた. これは二つのテンプレートによる集計・アラートの構造化により, 既存システムに比べてコマンド入力や整形の必要時間が削減できたためと考えられる. ただし, 必要時間の削減率 63.1%は, 入力回数の削減率 72.4%よりも低い. この要因は, 分析担当者が, 可視化結果からアラートの優先度を判断するために時間を要しているためと考えられる. 分析担当者からは, 構造化やテンプレートによって調査のためのコマンド作成の手間や整形する手間が減り, 提案システムは有用であるとされた. また, セキュリティ機器の検知ルールのアップデート時に, 誤検知が多発することがあり, すべてのセキュリティ機器のアラートが, 一元的に収集されていることで, アップデート後のアラート発生の挙動を簡単に調査できるようになったとのコメントも得られた.

4.5 結言

本章では, インシデント対応におけるアラート調査を迅速化する支援システムを提案した. 提案システムでは, アラートの形式がセキュリティ機器により異なる問題に対して, アラート調査に必要な項目をもとに統一的なアラート形式を定義し, セキュリティ機器のアラートをこの形式に変換することでアラートの形式を統一した. また, 調査のためのコマンドの作成を不要にするため, 調査に必要な標準的な集計処理を整理し, アラート発生時に事前に処理し, 必要であれば追加で集計処理を実行できるようにした. さらに, 発生アラートの流れを把握するためアラートを攻撃元とアラート間の時間間隔により, 2 段階に構造化させることで局所的な発生と継続的な発生を分かりやすく可視化した. 結果として, 提案システムは既存システムと比較して, 必要時間を平均 63.1%削減できることがわかった.

アラートの調査のさらなる迅速化としては, 通常発生していないユニークなアラート調査に対する支援が考えられる. 通常は発生しないユニークなアラートが同一の攻撃内に複数発生した場合, インシデントの可能性が高く詳細な調査が必要であることが多い. 現在の提案システムでは, 発生アラートは一覧で表示されているだけであるためユニークなアラートが同一の攻撃内に複数発生したか分かりづらい. これに対応するため, アラートの発生状況からユニークなアラートを自動的に判定したり, わかりやすく可視化したりする仕組みを検討する必要がある.

また，セキュリティ管理において，脆弱性が発見された際の影響度確認やシステム改善も日々のセキュリティ運用の負担となっている．特に近年のチャットボットシステムは，複数の OSS（Open Source Software）を組み合わせて出来ていることが多く，それら OSS の脆弱性の情報を日々確認し，アップデート時に問題が発生していないか確認するのに時間を要する．チャットボットで利用している OSS の脆弱性を自動的に収集し，それらの脆弱性情報から自動でチャットボットシステムをアップデートする必要がある．さらにアップデート時に，チャットボットに問題がでてないか確認する方式を検討する必要がある．例えば，アップデート時，過去の対話を擬似的に再現して同一の結果となるかなどで正常に動いているか確認する方法が考えられる．

第5章
結論

5.1　本研究のまとめ

本論文では，チャットボットシステムの開発と運用の効率化を提案し，これらの研究成果を以下の 4 章に分けて報告した.

第 1 章では，チャットボットシステムの重要性と概要について述べ，現状の課題を整理した. そして, それらの課題に対する関連研究について検討し, 研究方針を説明した.

第 2 章では, 対話入力用外部システムとチャットボットの並行導入における課題と解決手段を説明するため, 対話入力用外部システムにチャットボットのシナリオのステップや遷移を生成できる情報があり，その情報を取得できることを前提に，シナリオを生成する方式を提案した．従来方式では，シナリオ専用のデータ構造と対話入力用外部システムのデータ構造が異なるため, シナリオのすべての要素に対話入力用外部システムのデータを利用することは出来ず, 対話入力用外部システムのデータを利用できるのはシナリオ内の一部の要素にとどまっていた．これを解決するために，対話入力用外部システムのデータとシナリオの構造を対応させる変換定義, それにより定義される対話入力用外部システムのデータからシナリオを自動生成するシナリオ生成方法を提案した. 試行したチャットボットの適用において, 提案方式は手動でシナリオを作るのに比べて, シナリオの作成時間を 5 割以上削減できる見通しを得た.

第 3 章では，シナリオのテストにおける課題と解決手段を説明するため，シナリオのどこがどのような条件のときにシナリオとして正しいかを指定するシナリオ向きの動作条件定義方法と，動作条件違反の確認方法を提案した．動作条件定義方法では，シナリオの構成要素に対してシナリオの構造的な作成不良がない場合の動作の条件を定義する．動作条件違反の確認では，シナリオのルートをシナリオの構造のみから抽出し，指定した動作条件に準拠しているかを確認する. 提案方式はシナリオの作成で発生した 7 割の作成不良を発見した．さらに提案方式に加えて，手動での簡単なテストを組み合わせることで，9 割以上の作成不良を発見できた．これにより，テストの時間や作成不良の見逃しを削減することができ, 動作条件の記述時間も小さいことが有効性を確認で

きた.

第 4 章では，チャットボットの運用時，サイバー攻撃に対応するインシデント対応におけるアラート調査の課題と解決手段を説明するため，アラート調査支援システムを提案した．従来方式では，特定の攻撃調査を対象とした最適な可視化方式やアラート形式の変換方式などが提案されていたが．アラート調査全体を支援することが出来ておらず，インシデント対応を実運用するうえでの迅速化としては，不十分であった．これを解決するために，提案システムでは，アラートの形式がセキュリティ機器により異なる問題に対して，アラート調査に必要な項目をもとに統一的なアラート形式を定義し，セキュリティ機器のアラートをこの形式に変換することでアラートの形式を統一した．また，調査のためのコマンドの作成を不要にするため，調査に必要な標準的な集計処理を整理し，アラート発生時に事前に処理し，必要であれば追加で集計処理を実行できるようにした．さらに，発生アラートの流れを把握するためアラートを攻撃元とアラート間の時間間隔により，2 段階に構造化させることで局所的な発生と継続的な発生を分かりやすく可視化した．提案システムは既存システムと比較した結果，分析のためのコマンドの入力回数が 72.4%削減され，平均必要時間を 63.1%削減できた．また，分析担当者からは，構造化やテンプレートによってコマンド作成の手間や整形する手間が減り，提案システムは有用であるとされた.

5.2　今後の課題

チャットボットシステムのさらなる開発と運用の効率化のため，本研究を以下観点で進めるべきである.

・エスカレーションの自動化

チャットボットは，人と同じように対話できることが求められるが，現時点のチャットボットでは完全に人と同じように対話することはできない．このため，どうしても利用者の思うように対話が進まない状況が発生し，利用者がチャットボットの利用を途中で止めてしまうことがある．これを防ぐためには，利用者が思うように対話が進められなくなったことを検知し，コールセンターのオペレータにエスカレーションするなどの対応が必要になる[99], [100]．しかし，エスカレーションするタイミング（利用者が思ったように対話を進められていない状況の判断）は，利用者の特徴や，チャットボット

のユースケースやシナリオの構造などにより異なり，シナリオの作成者が適切な条件を指定するのは難しい．このため，シナリオの構造や対話の実績などから自動でエスカレーションする最適な条件を決める方式の検討が必要である．

・機密性が高い情報の保護の高度化

近年， EU 一般データ保護規則[101]などにより世界各国で個人情報など機密性が高い情報の保護が厳格化されている．一方でチャットボットは，医療や行政システムなどで利用が進んでおり，機密性が高い情報を取り扱うことが増えており，機密性が高い情報を適切に保護することが重要である[102]．チャットボットでは，オペレータによるサポートやシナリオ作成者のシナリオ改善などで，あとから対話内容を確認するために利用者との対話ログを一定期間保存しておくことが多い，対話時に機密性が高い情報が含まれている場合，セキュリティリスクとなる．利用者が機密性の高い情報を発話した場合，それを自動で検知し，暗号化する方式や，オペレータが対話ログを確認する際に，オペレータの権限により表示可能な情報を変える方法などが必要と考えられる．

謝　辞

本研究の全過程を通じて，終始懇切丁寧なるご指導とご鞭撻，格別のご配慮を賜りました大阪大学大学院情報科学研究科マルチメディア工学専攻 藤原融教授，薦田憲久招へい教授（大阪大学名誉教授）に深く感謝申し上げます.

本研究をまとめるにあたり，貴重なお時間を割いて頂き，丁寧なるご教示を賜りました大阪大学大学院情報科学研究科マルチメディア工学専攻 鬼塚真教授，松下康之教授に深く感謝申し上げます.

本研究をまとめるにあたり，親切なるご助言を賜りました大阪大学大学院情報科学研究科マルチメディア工学専攻 原隆浩教授，下條真司教授に深く感謝申し上げます.

本研究の機会と大学院博士後期課程に進学する機会を与えて頂くとともに，研究を進めるにあたりご配慮を賜りました（株）日立システムズ 前執行役員 佐川暢俊博士（現（株）セキュアブレイン代表取締役社長 兼 CEO），執行役員 赤津雅晴博士，本部長 田代卓氏に心より御礼申し上げます．また，上司として本研究と本論文をまとめる動機付けと機会を与えて頂くとともに，暖かいご指導とご鞭撻を頂きました，（株）日立システムズ 主管研究長 牧晋広博士，主管研究員 関口悦博氏，主管研究員 飯倉健氏，主任研究員 木村英志氏，主任研究員 多田直樹氏，研究員 角田朋氏に心から御礼申し上げます.

第 2 章，第 3 章の研究に関して，本研究の機会を与えて頂くとともに多大なるご支援を頂きました（株）日立システムズ 事業主管 工藤伸夫氏をはじめ，関係者の皆様に心より感謝いたします．また，共同研究者として様々なご討論ご助言を頂きました（株）日立システムズ 主任技師 津村直哉氏，五十嵐智氏に厚く御礼申し上げます.

第 4 章の研究にあたり，共同研究者として様々なご討論ご助言を頂きました（株）日立システムズ　前主任技師 大鳥朋哉氏，技師 小西幸洋氏に心から御礼申し上げます.

最後に，本論文の執筆にあたり，暖かく励まし支えてくれた家族，友人に心から感謝します.

参考文献

[1] S. Bisser, “Introduction to the Microsoft Conversational AI Platform,” in *Microsoft Conversational AI Platform for Developers: End-to-End Chatbot Development from Planning to Deployment*, Ed. Apress, pp. 1–24, 2021.

[2] 角田啓介, 後藤充裕, 北野孝俊, 中村浩司, 箕浦大祐, “チャットボットを用いたリモートワーク支援手法の提案,” 情報処理学会研究報告グループウェアとネットワークサービス, vol. 2017-GN-100, no. 38, pp. 1–7, 2017.

[3] J. Weizenbaum, “ELIZA - a computer program for the study of natural language communication between man and machine,” *Communications of the ACM*, vol. 26, no. 1, pp. 23–28, 1983.

[4] R. S. Wallace, R. Epstein, G. Roberts, and G. Beber, “The Anatomy of A.L.I.C.E.,” in *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*, Springer Netherlands, pp. 181–210, 2009.

[5] J. T. S. Quah and Y. W. Chua, “Chatbot Assisted Marketing in Financial Service Industry,” in *Proceedings of International Conference on Services Computing (SCC 2019)*, Italy, pp. 107–114, 2019.

[6] S. Gupta, D. Borkar, C. De Mello, and S. Patil, “An e-commerce website based chatbot,” *International Journal of Computer Science and Information Technologies*, vol. 6, no. 2, pp. 1483–1485, 2015.

[7] A. R. D. B. Landim, A. M. Pereira, T. Vieira, E. de B. Costa, J. A. B. Moura, V. Wanick, and E. Bazaki, “Chatbot design approaches for fashion E-commerce: an interdisciplinary review,” *International Journal of Fashion Design, Technology and Education*, vol. 15, no. 2, pp. 200–210, 2021.

[8] S. Holmes, A. Moorhead, R. Bond, H. Zheng, V. Coates, and M. Mctear, “Usability testing of a healthcare chatbot: Can we use conventional methods to assess conversational user interfaces?,” in *Proceedings of the 31st European Conference on Cognitive Ergonomics*, USA, pp. 207–214, 2019.

[9] D. Madhu, C. Jain, E. Sebastain, S. Shaji, and A. Ajayakumar, “A novel

approach for medical assistance using trained chatbot," in *Proceedings of the 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*, India, pp. 243–246, 2017.

[10] B. Ranoliya, N. Raghuwanshi, and S. Singh, "Chatbot for university related FAQs," in *Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, India, pp. 1525–1530, 2017.

[11] B. Heller, M. Proctor, D. Mah, L. Jewell, and B. Cheung, "Freudbot: An Investigation of Chatbot Technology in Distance Education," in *Proceedings of EdMedia + Innovate Learning 2005*, Canada, pp. 3913–3918, 2005.

[12] V. Kasinathan, M. H. A. Wahab, S. Z. S. Idrus, A. Mustapha, and K. Z. Yuen, "AIRA Chatbot for Travel: Case Study of AirAsia," *Journal of Physics: Conference Series*, vol. 1529, no. 2, p. 1–10, 2020.

[13] D. C. Ukpabi, B. Aslam, and H. Karjaluoto, "Chatbot Adoption in Tourism Services: A Conceptual Exploration," in *Robots, Artificial Intelligence, and Service Automation in Travel, Tourism and Hospitality*, Emerald Publishing Limited, pp. 105–121, 2019.

[14] 本田正美, "行政サービスに関わるチャットボット利用の可能性と課題," 情報処理学会研究報告ヒューマンコンピュータインタラクション（HCI）, vol. 2019-HCI-185, no. 5, pp. 1–5, 2019.

[15] I. Cantador, J. Viejo-Tardío, M. E. Cortés-Cediel, and M. P. Rodríguez Bolívar, "A Chatbot for Searching and Exploring Open Data: Implementation and Evaluation in E-Government," in *Proceedings of the 22nd Annual International Conference on Digital Government Research*, USA, pp. 168–179, 2021.

[16] 狩野芳伸, "コンピューターに話が通じるか : 対話システムの現在," 情報管理, vol. 59, no. 10, pp. 658–665, 2017.

[17] A. Rapp, L. Curti, and A. Boldi, "The human side of human-chatbot interaction: A systematic literature review of ten years of research on text-based chatbots," *International Journal of Human-Computer Studies*, vol. 151, p. 102630, 2021.

[18] S. Janarthanam, *Hands-On Chatbots and Conversational UI Development: Build chatbots and voice user interfaces with Chatfuel, Dialogflow, Microsoft Bot Framework, Twilio, and Alexa Skills*. Packt Publishing, 2017.

[19] M. Biswas, "Microsoft Bot Framework," in *Beginning AI Bot Frameworks: Getting Started with Bot Development*, Apress, pp. 25–66, 2018.

[20] 立花翔, LINE BOT を作ろう! Messaging API を使ったチャットボットの基礎と利用例. 翔泳社, 2017.

[21] 伊澤諒太, 井上研一, 江澤美保, 佐々木シモン, 羽山祥樹, 樋口文恵, 現場で使える! Watson 開発入門 Watson API、Watson Studio による AI 開発手法. 翔泳社, 2019.

[22] G. A. Kulkarni, *Building Chatbots with Microsoft Bot Framework and Node.js*. Manning Publications, 2019.

[23] 岩崎信也, 津村直哉, "身近になった対話システム：3. チャットボットサービスの変遷とそれを支える構成技術 ―シナリオ型チャットボットサービスの発展―," 情報処理, vol. 62, no. 10, pp. e12–e18, 2021.

[24] S. Shukla, L. Liden, S. Shayandeh, E. Kamal, J. Li, M. Mazzola, T. Park, B. Peng, and J. Gao, "Conversation Learner - A Machine Teaching Tool for Building Dialog Managers for Task-Oriented Dialog Systems," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Online, pp. 343–349, 2020.

[25] S. Bisser, "Building a Chatbot," in *Microsoft Conversational AI Platform for Developers: End-to-End Chatbot Development from Planning to Deployment*, Apress, pp. 177–238, 2021.

[26] S. Bisser, "Testing a Chatbot," in *Microsoft Conversational AI Platform for Developers: End-to-End Chatbot Development from Planning to Deployment*, Apress, pp. 239–251, 2021.

[27] S. Williams, *Hands-On Chatbot Development with Alexa Skills and Amazon Lex*. Packt Publishing, 2018.

[28] J. Bozic and F. Wotawa, "Planning-based Security Testing for Chatbots," in *Proceedings of the 21st International Multiconference Information Society*,

Slovenia, vol. E, pp. 23–26, 2018.

[29] 寺田真敏, 高田眞吾, 土居範久, “脆弱性対策情報データベース JVN の提案,” 情報処理学会論文誌, vol. 46, no. 5, pp. 1256–1265, 2005.

[30] 徳丸浩, 体系的に学ぶ安全な Web アプリケーションの作り方 第 2 版 脆弱性が生まれる原理と対策の実践. SB クリエイティブ, 2018.

[31] J. T. Luttgens, M. Pepe, K. Mandia, 政本憲蔵（監訳）, 凌翔太（監訳）, 山崎剛弥（監訳）, 高橋聡（訳）, 露久保由美子（訳）, 久保尚子（訳）, 舟津由美子（訳）, インシデントレスポンス 第 3 版. 日経 BP, 2016.

[32] S. Anso, 石川朝久（訳）, 詳解 インシデントレスポンス ―現代のサイバー攻撃に対処するデジタルフォレンジックの基礎から実践まで. オライリージャパン, 2022.

[33] J. Bollinger, B. Enright, M. Valites, 飯島卓也（監訳）, 小川梢（監訳）, 柴田亮（監訳）, 山田正浩（監訳）, 谷崎朋子（訳）, 実践 CSIRT プレイブック ―セキュリティ監視とインシデント対応の基本計画. オライリージャパン, 2018.

[34] JPCERT コーディネーションセンター, “インシデント対応とは？,” https://www.jpcert.or.jp/ir/ (accessed 2022).

[35] 藤田直行, “侵入検知に関する誤検知低減の研究動向,” 電子情報通信学会論文誌 B, vol. 89, no. 4, pp. 402–411, 2006.

[36] T. Pietraszek, “Using Adaptive Alert Classification to Reduce False Positives in Intrusion Detection,” in *Proceedings of International Workshop on Recent Advances in Intrusion Detection*, France, pp. 102–124, 2004.

[37] E. Adamopoulou and L. Moussiades, “An Overview of Chatbot Technology,” in *Proceedings of Artificial Intelligence Applications and Innovations*, Greece, pp. 373–383, 2020.

[38] P. Kucherbaev, A. Bozzon, and G. J. Houben, “Human-Aided Bots,” *IEEE Internet Computing*, vol. 22, no. 6, pp. 36–43, 2018.

[39] M. Yan, P. Castro, P. Cheng, and V. Ishakian, “Building a Chatbot with Serverless Computing,” in *Proceedings of the 1st International Workshop on Mashups of Things and APIs*, USA, pp. 1–4, 2016.

[40] P. Agrawal, T. Menon, A. Kam, M. Naim, C. Chouragade, G. Singh, R. Kulkarni,

A. Suri, S. Katakam, V. Pratik, P. Bansal, S. Kaur, A. Duggal, A. Chalabi, P. Choudhari, S. R. Satti, N. Nayak, and N. Rajput, “QnAMaker: Data to Bot in 2 Minutes,” in *Proceedings of the Web Conference 2020*, USA, pp. 131–134, 2020.

[41] 吉田尚水, 岩田憲治, 渡辺奈夕子, 小林優佳, 藤村浩司, “FAQ 集から自動で構築可能な QA 対話システム,” 人工知能学会研究会資料 言語・音声理解と対話処理研究会, vol. 87, pp. 113–114, 2019.

[42] 坂田亘, 田中リベカ, 黒橋禎夫, “公式ウェブサイトをベースにした QA チャットボットの自動構築,”言語処理学会 第 26 回年次大会発表論文集, pp. 327–330, 2020.

[43] A. Argal, S. Gupta, A. Modi, P. Pandey, S. Shim, and C. Choo, “Intelligent travel chatbot for predictive recommendation in echo platform,” in *Proceedings of 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, USA, pp. 176–183, 2018.

[44] C. V. S. Avila, A. B. Calixto, T. Rolim, W. Franco, A. D. P. Venceslau, V. Vidal, V. Pequeno, and F. F. D. Moura, “MediBot: An Ontology based Chatbot for Portuguese Speakers Drug’s Users,” in *Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS)*,Greece, pp. 25–36, 2019.

[45] 木下順史, 薦田憲久, 藤原融, “チャットボットを用いた社内情報サービス管理方式,” 第 19 回情報科学技術フォーラム（FIT2020）講演論文集, 第 4 分冊, pp. 169–172, 2020.

[46] D. Okanović, S. Beck, L. Merz, C. Zorn, L. Merino, A. van Hoorn, and F. Beck, “Can a Chatbot Support Software Engineers with Load Testing? Approach and Experiences,” in *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, USA, pp. 120–129, 2020.

[47] C. Webb and C. C. Limited, *Power Query for Power BI and Excel*. Apress, 2014.

[48] T. Leung, *Beginning PowerApps: The Non-Developers Guide to Building Business Mobile Applications*, Apress, 2017.

[49] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K.

Thiel, and B. Wiswedel, “KNIME - the Konstanz information miner: version 2.0 and beyond,” *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 26–31, 2009.

[50] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos, “Conceptual modeling for ETL processes,” in *Proceedings of the 5th ACM International Workshop on Data Warehousing and OLAP*, USA, pp. 14–21, 2002.

[51] J. Mayo, *Programming the Microsoft Bot Framework: A Multiplatform Approach to Building Chatbots*. Microsoft Press, 2017.

[52] M. Fewster, D. Graham, テスト自動化研究会（訳）, システムテスト自動化 標準ガイド. 翔泳社, 2014.

[53] P. Bourque and R. Fairley, *The Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*. IEEE Computer Society Press, 2014.

[54] D. Winkler, R. Hametner, T. Östreicher, and S. Biffl, “A framework for automated testing of automation systems,” in *Proceedings of the 2010 IEEE 15th Conference on Emerging Technologies Factory Automation (ETFA 2010)*, Span, pp. 1–4, 2010.

[55] M. Vasconcelos, H. Candello, C. Pinhanez, and T. dos Santos, “Bottester: Testing Conversational Systems with Simulated Users,” in *Proceedings of the XVI Brazilian Symposium on Human Factors in Computing Systems*, USA, pp. 1–4, 2017.

[56] M. B. dos Santos, A. P. C. C. Furtado, S. C. Nogueira, and D. D. Moreira, “OggyBug: A Test Automation Tool in Chatbots,” in *Proceedings of the 5th Brazilian Symposium on Systematic and Automated Software Testing*, USA, pp. 79–87, 2020.

[57] J. Bozic, O. A. Tazl, and F. Wotawa, “Chatbot Testing Using AI Planning,” in *Proceedings of the 2019 IEEE International Conference on Artificial Intelligence Testing (AITest)*, USA, pp. 37–44, 2019.

[58] K. Abdullah, C. P. Lee, G. Conti, J. A. Copeland, and J. Stasko, “IDS rainStorm: visualizing IDS alarms,” in *Proceedings of the IEEE Workshop on Visualization for Computer Security (VizSEC 05)*, USA, pp. 1–10, 2005.

[59] D. Inoue, M. Eto, K. Suzuki, M. Suzuki, and K. Nakao, “DAEDALUS-VIZ: Novel real-time 3D visualization for darknet monitoring-based alert system,” in *Proceedings of the 2012 ACM International Conference*, USA, pp. 72–79, 2012.

[60] Y. Livnat, J. Agutter, S. Moon, R. F. Erbacher, and S. Foresti, “A visualization paradigm for network intrusion detection,” in *Proceedings of the 6th Annual IEEE SMC Information Assurance Workshop*, USA, pp. 92–99, 2005.

[61] 井上和哉, 立岩佑一郎, 片山喜章, 高橋直久, “複数の IDS を用いたログ解析によるネットワーク診断システムについて,” マルチメディア, 分散協調とモバイルシンポジウム 2014 論文集, vol. 2014, pp. 461–475, 2014.

[62] 高田哲司, 小池英樹, “見えログ：情報視覚化とテキストマイニングを用いたログ情報ブラウザ,” 情報処理学会論文誌, vol. 41, no. 12, pp. 3265–3275, 2000.

[63] 江端真行, 小池英樹, “不正侵入調査を目的とした複数ログの時系列視覚化システム,” 情報処理学会論文誌, vol. 47, no. 4, pp. 1099–1107, 2006.

[64] T. Pietraszek and A. Tanner, “Data mining and machine learning—Towards reducing false positives in intrusion detection,” *Information Security Technical Report*, vol. 10, no. 3, pp. 169–183, 2005.

[65] 吉村尚人, 池上雅人, 長谷川智久, 原田隆史, 北谷浩, 森井昌克, “IDS アラートに対する誤検知削除方法の提案とその評価,” コンピュータセキュリティシンポジウム 2019 論文集, vol. 2019, pp. 467–473, 2019.

[66] S. O. Al-Mamory and H. Zhang, “Intrusion detection alarms reduction using root cause analysis and clustering,” *Computer Communications*, vol. 32, no. 2, pp. 419–430, 2009.

[67] Y. Meng and L. Kwok, “Adaptive False Alarm Filter Using Machine Learning in Intrusion Detection,” in *Proceedings of the Practical Applications of Intelligent Systems*, China, pp. 573–584, 2012.

[68] 小宅宏明, 宮地玲奈, 川口信隆, 重野寛, 岡田謙一, “機械学習によるネットワーク IDS の false positive 削減手法,” 情報処理学会論文誌, vol. 45, no. 8, pp. 2104–2112, 2004.

[69] 宮本貴朗, 泉正夫, 田村武志, 福永邦雄, “ネットワーク・サーバ運用監視支援シス

テム,” システム制御情報学会論文誌, vol. 15, no. 6, pp. 279–287, 2002.

[70] 竹森敬祐, 三宅優, 中尾康二, 菅谷史昭, 笹瀬巌, “Security Operation Center のための IDS ログ分析支援システム,” 電子情報通信学会論文誌 A, vol. J87-A, no. 6, pp. 816–825, 2004.

[71] 津田侑, 金谷延幸, 遠峰隆史, 神薗雅紀, 神宮真人, 高木彌一郎, 鈴木宏栄, “Nirvana 改によるライブネット分析,” 情報通信研究機構研究報告, vol. 62, no. 2, pp. 59–66, 2016.

[72] 上本悠貴, 岡村耕二, “STIX を用いた多様化する脅威情報の表現拡張に関する研究,” 情報処理学会シンポジウム, vol. 2018, no. 1, pp. 3–6, 2018.

[73] H. Debar, D. Curry, and B. Feinstein, “The Intrusion Detection Message Exchange Format (IDMEF),” *Request for Comments 4765*, 2007.

[74] 鈴木健太, 吉田健太郎, 大谷純, 道井俊介, データ分析基盤構築入門[Fluentd、Elasticsearch、Kibana によるログ収集と可視化]. 技術評論社, 2017.

[75] 日比野恒, Elastic Stack 実践ガイド[Logstash/Beats 編]. インプレス, 2020.

[76] 岩崎信也, 薦田憲久, 藤原融, “チャットボットにおけるシナリオの管理容易化のための動的実行方式, ” 第 19 回情報科学技術フォーラム（FIT2020）講演論文集, 第 4 分冊, O-001, pp. 161–164, 2020.

[77] 岩崎信也, 五十嵐智, 飯倉健, 津村直哉, 薦田憲久, 藤原融, “チャットボットと外部システムとの連携のためのシナリオ生成方式,” 電気学会論文誌 C, vol. 142, no. 8, pp. 919–928, 2022（掲載決定）.

[78] 岩崎信也, 薦田憲久, 藤原融, “チャットボットにおけるポリシーによるシナリオチェック方式, ” 情報処理学会第 84 回全国大会講演論文集, 第 4 分冊, 2G-03, pp. 441–442, 2022.

[79] 岩崎信也, 角田朋, 関口悦博, 小西幸洋, 大鳥朋哉, 薦田憲久, “不正侵入検知におけるセキュリティアラートのシグネチャ別発生量に着目した誤検知削減手法, ” コンピュータセキュリティシンポジウム 2018 論文集, pp. 441–442, 2018.

[80] S. Iwasaki, T. Kakuta, Y. Sekiguchi, Y. Konishi, T. Ohtori, and N. Komoda, “A Clustering - based Judgment Method of False Positive Alerts,” in *Proceedings of the 8th International Conference on Theory and Practice in Modern Computing 2019 (TPMC 2019)*, Portugal, pp. 174–180, 2019.

[81] 岩崎信也，角田朋，関口悦博，大鳥朋哉，薦田憲久, “アウトソース型セキュリティセンタにおけるインシデント対応迅速化のためのアラートログ可視化システム，” コンピュータセキュリティシンポジウム 2017 論文集，pp. 961–965, 2017.

[82] 岩崎信也, 角田朋, 関口悦博, 小西幸洋, 大鳥朋哉, 薦田憲久, “アウトソース型セキュリティセンタにおけるインシデント対応迅速化のためのアラート調査支援システム,” 情報処理学会論文誌, vol. 60, no. 4, pp. 1108–1118, 2019.

[83] D. D. McCracken and E. D. Reilly, “Backus-Naur form (BNF),” in *Encyclopedia of Computer Science*, John Wiley and Sons Ltd., pp. 129–131, 2003.

[84] D. Flanagan, 村上列（訳）, JavaScript 第 7 版. オライリージャパン, 2021.

[85] 山田耕嗣, 青沼亮太, 大嶋智子, 高橋徹, “kintone による学部生卒業研究指導システム構築、運用ならびに影響,” 情報システム学会全国大会論文集, vol. 13, p. e14, 2017.

[86] 掌田津耶乃, PowerApps ではじめるローコード開発入門 PowerFX 対応. ラトルズ, 2021.

[87] 相澤裕介, Kintone ファーストガイド 2022 年版―働き方改革を推進し、テレワークを実現!, カットシステム, 2021.

[88] J. D. Williams, E. Kamal, M. Ashour, H. Amr, J. Miller, and G. Zweig, “Fast and easy language understanding for dialog systems with Microsoft Language Understanding Intelligent Service (LUIS),” in *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Czech, pp. 159–161, 2015.

[89] S. Bird, E. Klein, E. Loper, 萩原正人（訳）, 中山敬広（訳）, 水野貴明（訳）, 入門 自然言語処理. オライリージャパン, 2010.

[90] 日吉龍, 不正侵入検知[IDS]入門 ―Snort＆Tripwire の基礎と実践. 技術評論社, 2004.

[91] 日本セキュリティオペレーション事業者協議会, “セキュリティ対応組織の教科書.” http://www.jnsa.org/result/2016/isog-j/ (accessed 2022).

[92] JPCERT コーディネーションセンター, “インシデントハンドリングマニュアル,” https://www.jpcert.or.jp/csirt_material/files/manual_ver1.0_20211130.pdf (accessed 2022).

[93] G. Blokdyk, *Unified Threat Management UTM A Complete Guide*. 5STARCooks, 2021.

[94] 水谷正慶, 白畑真, 南政樹, 村井純, “Session Based IDS の設計と実装,” 電子情報通信学会論文誌 B, vol. 88, no. 3, pp. 551–562, 2005.

[95] K. J. Cox and C. Gerg, *Managing Security with Snort & IDS Tools: Intrusion Detection with Open Source Tools*. O’Reilly Media, 2004.

[96] S. A. Miller, *Piwik Web Analytics Essentials*. Packt Publishing, 2012.

[97] Ruby サポーターズ, 改訂 2 版 パーフェクト Ruby. 技術評論社, 2017.

[98] 三輪賢一, 伊原智仁, 前川峻平, 内藤裕之, 福井隆太, Palo Alto Networks 構築実践ガイド 次世代ファイアウォールの機能を徹底活用. 技術評論社, 2015.

[99] A. Følstad and M. Skjuve, “Chatbots for customer service: user experience and motivation,” in *Proceedings of the 1st International Conference on Conversational User Interfaces*, USA, pp. 1–9, 2019.

[100] C. Abbet, M. M’hamdi, A. Giannakopoulos, R. West, A. Hossmann, M. Baeriswyl, and C. Musat, “Churn Intent Detection in Multilingual Chatbot Conversations and Social Media,” in *Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018)*, Belgium, pp. 161–170, 2018.

[101] P. Voigt and A. Bussche, *The EU General Data Protection Regulation (GDPR)*. Springer Cham, 2017.

[102] S. T. Lai, F. Y. Leu, and J. W. Lin, “A Banking Chatbot Security Control Procedure for Protecting User Data Security and Privacy,” in *Proceedings of the Advances on Broadband and Wireless Computing, Communication and Applications*, Belgium, pp. 561–571, 2019.