文章编号:1671 - 4598(2026)06 - 0139 - 08 DOI:10.16526/j.cnki.11-4762/tp.2026.06.018 中图分类号:TP312 文献标识码:A

基于 Java 的针织控制系统设计与实现

张团善，吴 妙，韩起立
（西安工程大学 机电工程学院，西安 710613）

摘要：针对针织机操作过程中可视化与智能控制的需求，设计并实现了一套集成编织指令解析、状态管理、异常处理与二维可视化的针织机操作控制系统；系统采用模块化架构，包含指令解析、操作状态管理、异常处理、二维渲染与图形用户界面等核心模块；通过正则表达式与状态机方法实现对多种编织指令的自动解析与执行，支持针数设置、编织、转移、叠针、跳针、移位、引纱、退纱及反向编织等多种操作；系统利用 JavaFX 与 Swing 混合技术，开发了支持二维动画的可视化界面，实现了针床、针及纱线等结构的动态渲染与交互，能够实时反映操作序列对针床状态的影响；实验结果表明，该系统能够准确解析并执行复杂编织流程，二维可视化响应延迟低于 100 ms，指令解析速度达到每秒 1 000 余条指令的处理能力；系统解析准确率达到 99.8%，执行效率相比传统手工操作提升约 80%；稳定性测试验证了系统在连续运行 8 h、处理 1 000 余条编织指令场景下的可靠性，无内存泄漏或性能衰减现象，极大地提升了针织工艺的可视化表达与智能控制能力；该方法为针织机数字化、智能化升级提供了有效的技术支撑。

关键词：针织机；编织指令解析；状态管理；智能控制；人机交互

Design and Implementation of Knitting Control System Based on Java

ZHANG Tuanshan, WU Miao, HAN Qili
(School of Mechanical and Electrical Engineering, Xi'an Polytechnic University, Xi'an 710613, China)

Abstract: To address the needs of visualization and intelligent control in knitting machine operations, an integrated operation control system for knitting machines is designed and implemented, which includes instruction parsing, state management, exception handling, and 2D visualization. The system adopts a modular architecture, including core modules such as instruction parsing, operation state management, exception handling, 2D rendering, and graphical user interface. By using regular expressions and finite-state machine methods, the system automatically parses and executes diverse knitting instructions, supporting multiple operations such as stitch setting, knitting, transferring, stacking, skipping, shifting, yarn feeding, yarn retraction, and reverse knitting. The system employs a hybrid JavaFX and Swing framework to develop a visualization interface with 2D animation, achieves dynamical rendering and interaction for structures such as needle beds, needles, and yarns, and reflects the impact of operation sequences on needle bed state in real time. Experimental results demonstrate that the system accurately parses and executes complex knitting processes, which achieves the 2D visualization response delay by below 100 ms, the instruction parsing speed by over 1000 instructions per second, and the accuracy of system parsing by 99.8%, increasing the execution efficiency by about 80% compared to traditional manual operations. Stability tests verify the system's reliability under the conditions of running for 8 hours and processing over 1 000 weaving instructions, without any memory leaks or performance degradation, which significantly enhances the visual expression and intelligent control capabilities of knitting techniques, providing an effective technical support for the digital and intelligent upgrading of knitting machines.

Keywords: knitting machine; knitting instruction parsing; state management; intelligent control; human-machine interaction

0 引言

近年来，随着纺织工业智能化与数字化进程的加快，针织机的自动化控制与可视化技术逐渐成为学术界和工业界关注的研究热点。现代纺织制造业对生产效率、产品质量以及柔性化生产提出了更高的要求，推动

收稿日期:2025 - 07 - 08； **修回日期**:2025 - 08 - 19。
基金项目:国家自然科学基金(51735010)；西安现代智能纺织设备重点实验室资助项目(2019220614SYS021CG043)。
作者简介:张团善(1971 -),男,博士,教授。
引用格式:张团善,吴 妙,韩起立. 基于 Java 的针织控制系统设计与实现[J]. 计算机测量与控制,2026,34(6):139 - 146.

了针织设备向智能化、信息化方向不断发展。在这一背景下，针织机的操作流程、状态监控、故障诊断等环节的信息透明度和交互性显得尤为重要。然而，传统针织工艺在实际应用中普遍存在操作流程复杂、信息传递不畅、状态监控不直观、故障诊断效率低下等问题，导致生产过程难以实现精细化管理和智能化调度，难以满足当前智能制造对高效、可视、可控生产的需求。

国内外学者在针织机智能控制系统和领域特定语言（DSL）方面开展了大量研究。在针织机控制系统方面，有基于递归模式的针织算法描述方法[9]，为针织图案的数学建模奠定了基础；开发了针织图案设计的 CAD 模块，实现了基本的图案生成功能[11]；还有学者提出了集成化的针织服装设计模型，但在指令解析和实时可视化方面存在局限性[18]。国内学者研究了全自动电脑横机的花型准备和控制系统，主要关注硬件控制层面[6]；还有学者针对嵌入式电脑横机进行了花型数据的编译处理研究，但缺乏完整的可视化界面[19]；在远程协作方面，有学者做了基于互联网的经编针织物 CAD 系统，但在指令解析的灵活性和异常处理机制方面仍有不足[5]。

在针织领域特定语言研究方面，Bernasconi 提出了针织递归模式的形式化描述方法，为针织 DSL 的理论基础做出了重要贡献。然而，现有研究大多集中在理论层面，缺乏完整的语言实现和可视化支持。在技术架构方面，现有系统普遍存在以下问题：1）指令解析功能单一，缺乏对复杂针织操作的全面支持；2）状态管理机制不完善，难以实现实时状态监控和异常检测；3）可视化表现力有限，无法直观展示编织过程的动态变化；4）异常处理机制不健全，缺乏精确的错误定位和友好的用户反馈；5）系统架构耦合度高，扩展性和维护性较差。

为解决上述难题，本文面向针织机操作过程的智能可视化需求，设计并实现了一套集成编织指令解析、状态管理、二维可视化控制、异常处理等多项功能于一体的针织机智能控制与可视化系统。该系统采用模块化结构设计，核心包括状态管理系统、指令解析模块、二维渲染引擎以及图形用户界面等，能够支持多种编织操作的自动解析与高效执行。系统通过 JavaFX 与 Swing 混合技术，实现了针床、针、纱线等关键结构的动态二维可视化，极大地提升了操作过程的直观性和交互体验。同时，系统具备完善的异常处理机制，能够对操作过程中的异常情况进行及时反馈和友好提示，保障了系统的稳定性和易用性。

本研究预期通过该系统的开发与应用，显著提升针织工艺的可视化表达能力与智能控制水平，为针织机的数字化升级和智能制造转型提供坚实的技术支撑和理论参考，推动纺织行业向高端智能制造迈进[1-3]。

1 针织 DSL 系统设计

1.1 系统总体架构

本针织 DSL 系统整体架构采用模块化、分层式设计，充分体现了高内聚、低耦合的现代软件工程思想。系统主要由 4 大核心功能模块组成，分别为：编织指令输入与解析模块、操作对象生成与业务逻辑处理模块、系统状态管理与可视化渲染模块，以及人机交互与异常处理模块。各模块之间通过明确定义的数据流和接口进行协作，既保证了系统的灵活扩展性，也提升了运行的稳定性和可维护性。

编织指令输入与解析模块负责接收用户通过图形界面输入或导入的编织程序代码，结合内置的 DSL 指令文档和语法规则，对输入内容进行语法分析和合法性校验，并将其转化为标准化的操作对象。随后，操作对象生成与业务逻辑处理模块对解析得到的操作对象进行封装，依次驱动核心业务逻辑的执行，包括针织操作的自动化调度、设备参数的配置等。该模块还支持对操作指令的灵活扩展，能够适应多样化的编织工艺需求。

在此基础上，系统状态管理与可视化渲染模块实时维护针床、针、纱线等关键部件的状态信息，并通过 JavaFX 与 Swing 混合技术实现动态的二维可视化展示。该模块不仅能够直观反映编织过程的每一步操作，还支持动画播放和结果回溯，极大提升了用户的交互体验和工艺理解能力。与此同时，人机交互与异常处理模块为用户提供友好的操作界面和完善的异常反馈机制，能够对系统运行过程中出现的各类异常进行捕获、处理与提示，保障系统的稳定性和易用性。

系统整体基于 Java 平台开发，采用 Maven 进行项目依赖管理，并通过 module-info.java 实现各功能模块的封装与隔离，确保了架构的清晰性和可扩展性。各模块之间通过标准化接口和数据流进行通信，既支持独立开发与测试，也便于后续功能的拓展和维护。通过上述架构设计，针织 DSL 系统能够高效支持编织工艺的智能解析、动态可视化与人机交互，为针织机数字化与智能化升级提供坚实的技术基础。系统整体架构如图 1 所示。

在语言解析层，KnittingParser 类负责针织 DSL 的词法分析和语法解析，将文本形式的针织指令解析为 KnittingOperation 对象，支持 init、knit、xfer、tuck、miss、rack 等核心指令，并通过正则表达式进行严格的语法检查，针织指令的枚举如表 1 所示。

核心业务处理层由 KnittingInterpreter 类实现，作为解释器核心，负责指令的语义分析和执行控制，通过状态机模式管理编织过程，确保指令执行的正确性和顺序性。

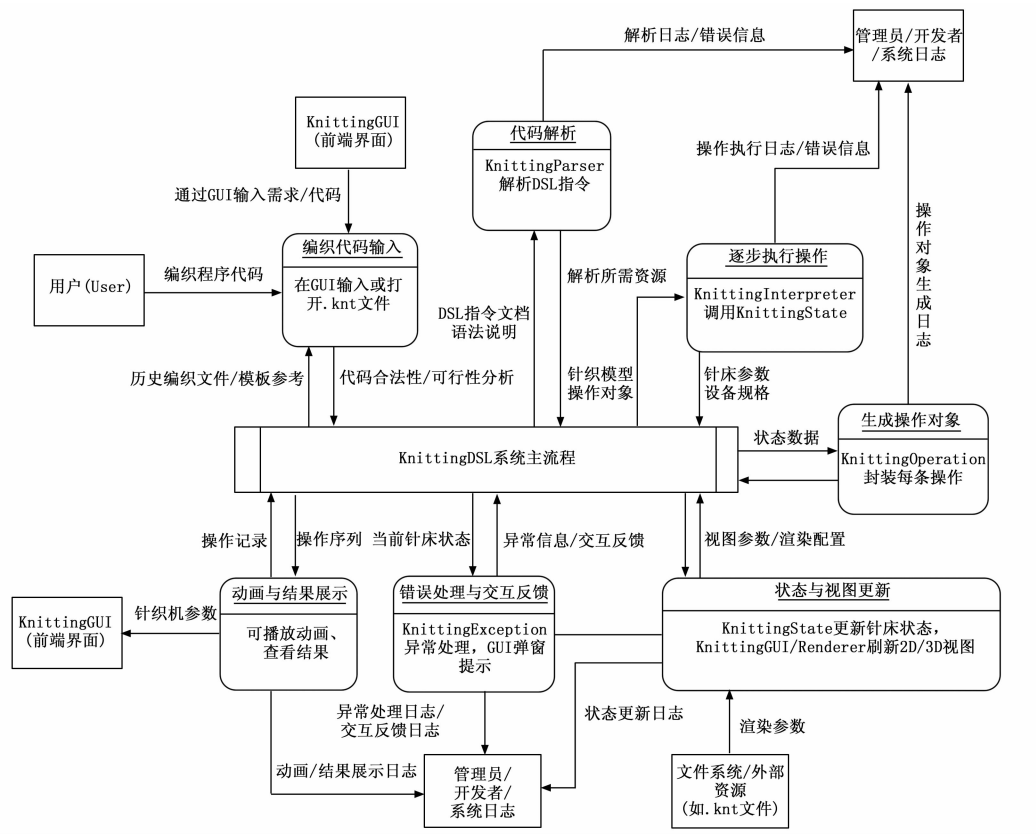

图 1　DSL 系统总体架构

表 1　针织指令枚举

操作指令	解释	作用
Init	针数	作为第一条有效指令，用于初始化前后针床的针位数量
Knit	编织	执行基本的编织操作，在指定针位上形成线圈
Xfer	转移	在前后针床之间转移线圈，用于实现复杂的针织结构如提花
Tuck	叠针	在已有线圈上叠加新的线圈，用于特殊的针织效果，如浮线
Miss	跳针	跳过指定针位，不进行编织
Rack	针床移位	控制后针床相对于前针床的水平移动
Inhook	引纱	引入纱线，准备进行编织操作
Outhook	退纱	退出纱线，结束编织过程
Reverse	反向编织	实现反向编织效果

在状态管理层，KnittingState 类维护双针床的实时状态，包括前后针床的线圈分布、针床移位位置、当前针数和引纱状态等关键信息，通过数组数据结构实现高效的状态存取。

用户交互层采用 KnittingGUI 类构建图形界面，基于 Java Swing 技术实现，提供代码编辑、实时预览、错误提示等功能，并通过 KnittingRenderer 类实现针织过程的 2D 可视化。

系统采用异常处理机制，通过 KnittingException 类实现精确的错误定位和友好的错误提示。整个系统遵循面向对象设计原则，采用工厂模式、单例模式和观察者模式等设计模式，确保了系统的可扩展性和可维护性。

在性能优化方面，系统通过 JavaFX 属性配置和线程管理，实现了流畅的用户交互体验。测试框架采用 JUnit，通过 KnittingTest 类实现了完整的单元测试和集成测试，覆盖了核心功能和边界情况，保证了系统的稳定性和可靠性[4-7]。

1.2　针织领域概念分析

针织领域的核心概念可以从针织机的物理结构和操作行为两个维度进行分析。在物理结构方面，本系统采用双针床结构作为基础模型，包括前针床（Front Bed）和后针床（Back Bed），每个针床上的针位使用 f0-fn 和 b0-bn 进行编号，这种编号方式直观地反映了针织机的实际结构。针床状态由 KnittingState 类维护，通过整数数组 frontBed [] 和 backBed [] 分别记录前后针床上每个针位的状态，其中 0 表示无线圈，1 表示有线圈，−1 表示特殊状态。针床之间的相对位置通过 rack 值（范围为−4～4）来表示，这反映了实际针织过程中针床的横向移动能力。在操作行为方面，系统定义了一组基本操作类型（OperationType），包括：编织（Knit）、叠针（Tuck）、跳针（Miss）、转移（Xfer）、反向编织（Reverse）和提花（Jacquard）等。这些操作都需要在引

纱（Inhook）状态下进行，并可以通过退纱（Outhook）结束编织过程。每个操作都包含方向属性（Direction，用+/－表示向右/向左）、位置信息（Position，如 f0、B5 等）和操作数量（Count）。系统通过 Knitting-Operation 类对这些操作进行封装，确保操作参数的合法性，例如：针位格式必须符合［fb］\\d+的正则表达式模式，移位值必须在有效范围内。在状态转换方面，系统实现了严格的状态检查机制，通过 KnittingException 处理异常情况，如：在未引纱状态下进行编织操作、向已有线圈的针位转移等非法操作都会触发相应的异常。这种领域概念的形式化描述为 DSL 的设计和实现提供了清晰的理论基础，同时也确保了系统行为与实际针织过程的一致性[8-11]。

1.3 DSL 语言设计

1.3.1 基本语法规则

针织 DSL 采用简单直观的语法结构，每行指令代表一个独立的针织操作。指令格式采用" 操作名 参数列表" 的形式，参数之间使用空格分隔。语言支持单行注释，使用双斜线（//）标识。基本语法规则如下：

1）初始化指令：init ＜针数＞。

用于设置针床的初始针数，必须作为程序的第一条有效指令。

2）引纱控制。

inhook：引入纱线；

outhook：退出纱线。

3）编织操作：＜操作类型＞ ＜方向＞ ＜位置＞ ＜数量＞。

操作类型：knit（编织）、tuck（叠针）、miss（跳针）。

方向：＋表示向右，－表示向左。

位置：*f* 数字表示前针床，*b* 数字表示后针床。

数量：表示连续操作的针数。

4）转移操作：xfer ＜源位置＞ ＜目标位置＞。

用于在前后针床之间转移线圈，位置格式为 f/b 数字。

5）针床移位：rack ＜偏移量＞。

控制后针床的水平移动，偏移范围为［－4，4］。

6）反向编织：reverse ＜方向＞ ＜位置＞ ＜数量＞。

实现反向编织效果。

7）提花编织：jacquard ＜位置＞ ＜针数＞ ＜图案＞。

实现提花编织效果，图案使用 0/1 序列表示。

1.3.2 指令系统设计

指令系统的设计采用分层架构，通过 KnittingOperation 枚举类型定义了 9 种基本操作类型：KNIT、XFER、TUCK、MISS、RACK、INHOOK、RELEASEHOOK、STITCH _ NUMBER 和 JACQUARD。每种操作类型都具有特定的参数验证规则和执行逻辑。指令系统的核心特征包括：

1）状态依赖控制：所有针织操作（除 init 和 inhook 外）都要求纱线处于引入状态，确保操作的有效性。例如：编织操作必须在执行 inhook 之后才能进行，这种依赖关系通过 KnittingInterpreter 中的 hasYarn 标志进行控制。

2）参数验证机制：每个操作都有严格的参数格式要求，通过正则表达式进行匹配验证。如针位格式必须符合" ［fb］\\d+" 模式，方向必须是" ［＋－］"，数量必须为正整数。

3）系统通过 KnittingState 类构建了针位操作的状态机模型，该模型具备：操作指令的时序存储、针位状态的版本管理、历史操作的追溯验证 3 大核心功能。

1.3.3 错误处理机制

错误处理机制采用多层异常处理策略，通过自定义的 KnittingException 类实现精确的错误定位和提示如：

```
public class KnittingException extends RuntimeException {
    private static final long serialVersionUID = 1L;
    private int lineNumber;
    public KnittingException(int line,String message) {
      super("Line " + line + ":" + message);
      this.lineNumber = line;
    }
    public int getLineNumber() {
      return lineNumber;
      }
}
```

运行时验证：

1）针床范围检查：确保操作不超出针床边界。

2）状态冲突检测：防止向已有线圈的针位转移新的线圈。

3）移位范围控制：确保 rack 指令的移位值在有效范围内。

4）操作序列验证：确保操作顺序符合针织工艺要求。

5）异常恢复机制：提供清晰的错误提示信息，包含行号和具体原因，支持操作回滚，保证针床状态的一致性，通过 GUI 界面实时反馈错误信息，通过简洁的语法结构和严格的类型检查，确保了针织程序的可靠性和可维护性。错误处理机制保证系统在面对异常情况时能够及时地处理并提供有效的反馈，提高了系统的健壮性和友好性。整个语言设计遵循" 易学易用" 的原则，既满足了专业针织工艺的需求，又降低了使用门槛，为针织图案设计提供了有力的工具支持。通过这种设计，针织 DSL 不仅实现了对针织操作的精确描述，还建立了一个完整的针织程序验证体系，为后续的可视化展示和工艺优化奠定了基础[12-15]。

2　针织 DSL 解释器实现

2.1　词法分析与语法解析

DSL 系统的词法分析与语法解析模块主要由 KnittingParser 类实现，采用正则表达式进行模式匹配和指令解析。系统定义了一组完整的指令模式，包括初始化（init）、编织（knit）、转移（xfer）、叠针（tuck）、跳针（miss）、针床移位（rack）、引纱（inhook）、退纱（outhook）、反向编织（reverse）和提花编织（jacquard）等操作。KnittingParser 类中定义了对应的正则表达模式，如：

```
public class KnittingParser {
    private static final Pattern INIT_PATTERN =
        Pattern.compile("^\s*init\s+(\d+)\s*");
    private static final Pattern KNIT_PATTERN =
        Pattern.compile("^\s*knit\s+([+-])\s+([fb]\d+)\s+(\d+)\s*(;\meta=\d+)? \s*");
    private static final Pattern XFER_PATTERN =
        Pattern.compile("^\s*xfer\s+([fb]\d+)\s+([fb]\d+)\s*");
    private static final Pattern TUCK_PATTERN =
        Pattern.compile("^\s*tuck\s+([+-])\s+([fb]\d+)\s+(\d+)\s*");
    private static final Pattern MISS_PATTERN =
        Pattern.compile("^\s*miss\s+([+-])\s+([fb]\d+)\s+(\d+)\s*");
    private static final Pattern RACK_PATTERN =
        Pattern.compile("^\s*rack\s+(-? \d+)\s*");
    private static final Pattern INHOOK_PATTERN =
        Pattern.compile("^\s*inhook\s*");
    private static final Pattern OUTHOOK_PATTERN =
        Pattern.compile("^\s*outhook\s*");
    private static final Pattern REVERSE_PATTERN =
        Pattern.compile("^\s*reverse\s+([+-])\s+([fb]\d+)\s+(\d+)\s*");
    private static final Pattern JACQUARD_PATTERN =
        Pattern.compile("^\s*jacquard\s+([fb]\d+)\s+(\d+)\s+([01]+)\s*",Pattern.CASE_INSENSITIVE);
```

解析器通过 parse 方法对每行指令进行解析，将文本指令转换为 KnittingOperation 对象。以编织操作为例，其解析实现如下：

```
private KnittingOperation parseKnitOperation(Matcher matcher,int lineNumber) {
    try {
        KnittingOperation op = new KnittingOperation(KnittingOperation.OperationType.KNIT);
        op.setDirection(matcher.group(1)); // 获取方向(+/-)
        op.setPosition(matcher.group(2)); // 获取位置(f数字/b数字)
        op.setCount(Integer.parseInt(matcher.group(3))); // 获取数量
        String metaData = matcher.group(4);
        if(metaData != null) {
            String metaValue = metaData.substring(metaData.indexOf("=") + 1);
            op.setMetadata(Integer.parseInt(metaValue));
        }
        return op;
    } catch(Exception e) {
        throw new KnittingException(lineNumber,"编织指令解析错误:" + e.getMessage());
    }
}
```

解析得到的操作对象包含了完整的指令信息，定义如下：

```
private OperationType type;
private String direction;      // +/-
private String position;       // f数字/b数字
private String fromPosition;   // 转移起始位置
private String toPosition;     // 转移目标位置
private int count;             // 操作数量
private int metadata = -1;     // 元数据值
private String comment;        // 注释
```

该解析系统能够准确识别指令类型、参数和注释，为后续的指令执行提供了结构化的数据支持。基于形式语言理论构建的语法约束体系与容错处理架构，为针织程序提供了从语法解析到运行时监控的全流程质量保障。测试结果表明，该解析器可以正确处理所有合法的针织指令，并对不合法指令给出准确的错误提示[16-18]。

2.2　状态管理

2.2.1　针床状态模型

针织 DSL 系统采用双针床状态模型设计，通过 KnittingState 类实现对前后针床状态的实时管理。该模型使用两个整型数组 frontBed 和 backBed 分别表示前后针床，数组中的每个元素代表对应针位的状态：0 表示无线圈，1 表示有线圈。针床状态模型还包含了针床移位位置 rackPosition、总针数 stitchNumber 和引纱状态 hooked 等关键属性。以下是针床状态模型的核心实现：

```
private int[] frontBed;        // 前针床
private int[] backBed;         // 后针床
private int rackPosition;      // 针床移位位置
private int stitchNumber;      // 针数
private boolean hooked;        // 引纱状态

public KnittingState(int needleCount) {
    frontBed = new int[needleCount];
```

```
        backBed = new int[needleCount];
        frontBedOperations = new OperationType[needleCount][10];// 每个针位最多记录 10 个操作
        backBedOperations = new OperationType[needleCount][10];// 每个针位最多记录 10 个操作
        rackPosition = 0;
        stitchNumber = 0;
        hooked = false;
        printMessage("\n 初始化针床:");
        printMessage("针床长度:" + needleCount);
        printBedStatus();
    }
```

状态模型支持针织操作的执行和状态验证，例如在执行转移操作时，系统会检查源针位和目标针位的状态，该状态模型通过严格的状态检查和转换机制，确保了针织操作的正确性和可靠性，为整个 DSL 系统提供了坚实的状态管理基础。

2.2.2 操作执行机制

操作执行由 KnittingInterpreter 类控制，针对每种操作类型实现专门的执行方法，如下：

```
    private void executeOperation(KnittingOperation op) {
        try {
            switch(op.getType()) {
                case KNIT:
                    executeKnit(op);
                    break;
                case XFER:
                    executeTransfer(op);
                    break;
                case TUCK:
                    executeTuck(op);
                    break;
                case MISS:
                    executeMiss(op);
                    break;
                case RACK:
                    executeRack(op);
                    break;
                case INHOOK:
                    executeInhook(op);
                    break;
                case RELEASEHOOK:
                    executeReleasehook(op);
                    break;
                case STITCH_NUMBER:
                    executeStitchNumber(op);
                    break;
                case REVERSE:
                    executeReverse(op);
                    break;
                case JACQUARD:
                    executeJacquard(op);
                    break;
                default:
                    handleError("未知的操作类型",new Exception(op.getType().toString()));
            }
        } catch(Exception e) {
            handleError("执行操作失败",e);
        }
    }
```

针织 DSL 解释器采用策略模式实现操作执行机制，通过 executeOperation 方法作为中央调度器，将解析后的针织指令分发到对应的执行策略。该机制支持十种核心针织操作类型，包括基础编织操作（编织、转移、叠针、跳针）、机器控制操作（引纱、退纱、针床移位）以及高级功能（反向编织、初始化设置等）。

针织操作的具体执行由 KnittingState 类负责，该类维护了前后针床的状态数据，并实现了各类操作的底层逻辑。

GUI 会实时将操作执行过程中的状态变化反馈给用户。系统支持的基本操作包括编织（knit）、转移（xfer）、叠针（tuck）、跳针（miss）、针床移位（rack）等，每个操作都有严格的前置条件检查和状态更新逻辑。基于模块化设计原则构建的系统架构，不仅支持操作功能的无缝扩展，还通过标准化的接口设计和异常处理机制，显著提升了操作执行的稳定性和系统的可维护水平[19-20]。

2.3 异常处理实现

针织 DSL 采用分层异常处理架构，通过自定义异常类 KnittingException 实现统一的错误管理机制。该架构包含 3 个核心层次：语法解析层、语义验证层和执行操作层，每层都配备相应的异常捕获和处理机制，异常处理流程如图 2 所示。

在 KnittingInterpreter 类中实现了 handleError 方法，作为统一的错误处理入口，该方法实现了多通道错误输出，同时向控制台和 GUI 界面报告错误信息，并统一将系统异常转换为领域特定的 KnittingException。如：

```
    private void handleError(String message,Exception e) {
        String errorMessage = message + ":" + e.getMessage();
        System.err.println(errorMessage);
        KnittingGUI.getInstance().appendText("\n 错误:" + errorMessage);
        if(!(e instanceof KnittingException)) {
            throw new KnittingException(0,errorMessage);
        } else {
```

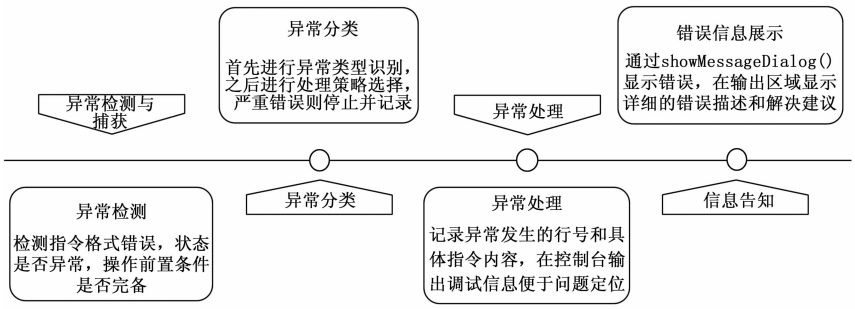

图 2　异常处理流程

```
            throw(KnittingException)e;
        }
    }
```

系统针对不同类型的异常设计了专门的处理策略。在语法解析层，KnittingParser 类使用正则表达式模式匹配算法验证指令格式，如 KNIT _ PATTERN、XFER _ PATTERN 等，确保指令参数符合预定义格式。当检测到格式错误时，系统抛出包含行号信息的 KnittingException，便于用户定位问题。在语义验证层，KnittingState 类实现了状态一致性检查算法，通过 validateState 方法验证针床状态的有效性，包括针位范围检查、状态值验证和操作前置条件检查。对于转移操作，系统采用双重验证机制：首先检查源针位是否有线［fromBed（fromPos）＝＝0］，然后验证目标针位是否已有线［toBed（toPos）！＝ 0］，确保操作的逻辑正确性。在执行操作层，系统实现了状态回滚机制，当操作执行过程中发生异常时，能够恢复到操作前的安全状态。对于 GUI 界面异常，系统采用备选渲染策略，当 3D 渲染初始化失败时自动切换到 2D 渲染模式，确保用户界面的可用性。此外，系统还实现了线程安全的异常处理机制，使用 SwingUtilities. invokeLater 和 Platform. runLater 确保 GUI 更新操作的线程安全性。通过这些分层的异常处理机制，系统能够在各种异常情况下保持稳定运行，为用户提供可靠的错误恢复和状态一致性保证。

3　针织仿真

3.1　2D 可视化设计

针织 DSL 系统的 2D 可视化仿真模块基于 Java Swing 技术开发，旨在为用户提供直观、实时的编织过程展示。该模块通过自定义绘制组件，动态反映针床、针、纱线等关键部件的状态变化，实现了对针织工艺全过程的二维仿真。系统在核心的 KnittingGUI 类中集成了双针床的实时状态显示功能，能够根据编织指令的执行进度，动态刷新每一根针的状态和操作轨迹。

在可视化设计方面，系统采用 20 像素为单位的针距布局，通过重写 paintComponent 方法实现对针床状态的高效渲染。针对不同类型的针织操作（如编织、叠针、跳针及转移等），系统为每种操作定义了独特的图形符号，例如实线、虚线、三角形及菱形等，并结合红色、蓝色、绿色及灰色等多种颜色进行区分，极大提升了操作的辨识度和可读性。每当用户执行编织指令或播放动画时，系统会根据当前操作序列，实时更新 2D 视图，确保用户能够清晰地观察到每一步操作对针床状态的影响。

此外，2D 可视化模块充分利用 Java 2D 图形库的抗锯齿和高质量渲染选项，有效提升了图形显示的平滑度和细节表现力。用户不仅可以通过界面直观地查看编织过程，还能获得实时的操作反馈和状态提示，便于及时发现和纠正操作中的异常。整体而言，该 2D 仿真模块为针织工艺的数字化表达和交互体验提供了有力支撑，是系统实现智能可视化控制的重要组成部分，如图 3 所示。

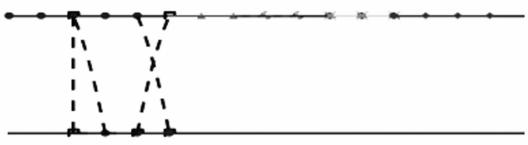

图 3　KnittingGUI 类中的 2D 动态仿真

3.2　动态仿真实现

针织 DSL 的 2D 动态仿真模块基于 Swing Timer 机制实现，如图 3 为针织 DSL 系统的用户交互界面，点击播放 2D 动画即可实现在 2D 图上实现动态仿真，通过逐步骤动画播放技术展示针织操作的执行过程。该模块采用状态驱动的动画控制策略，将静态的针织操作序列转换为动态的可视化展示。

组件功能：

1）play2DButton：用户交互入口，触发动画播放；

2）play2DTimer：Swing 定时器，控制动画播放节奏；

3）play2DStepIndex：动画进度索引，记录当前播放位置。

4　结束语

本研究成功设计并实现了一个完整的针织领域特定语言（DSL）系统，通过系统性的架构设计和模块化实现，验证了领域特定语言在专业领域应用中的核心原理。研究表明，通过将针织工艺的复杂操作抽象为简洁的指令序列，能够有效降低用户的学习成本，提高编程效率。系统采用的分层架构设计（语法解析层、语义验证层、执行操作层）为其他专业领域的 DSL 开发提供了可复用的设计模式。实验结果表明，该针织 DSL 系统能够准确解析和执行十种核心针织操作，验证了领域

特定语言在专业工艺控制中的可行性和有效性。

参考文献：

[1] 王丹阳. 针织行业运行质量明显改善 [N]. 北京：中国纺织社，2017－09－18 (02).

[2] 张永超，丛洪莲，张爱军. 纺织 CAD 技术进展与发展趋势 [J]. 北京：中国纺织信息中心，2015 (7)：40－43.

[3] 王 敏，刘斯年，金万慧. 无缝针织技术及无缝针织机的历史与发展 [J]. 北京：中国纤维检验局，2020，41 (1)：126－127.

[4] 岑 凌. 2018 中国国际纺织机械展览会暨 ITMA 亚洲展览会无缝内衣圆机述评 [J]. 针织工业，2018，46 (11)：9－10.

[5] 汤梦婷，蒋高明，王 薇，等. 基于互联网的经编针织物 CAD 系统开发与实现 [J]. 纺织学报，2018，39 (10)：143－148.

[6] 伍 昌. 全自动电脑横机花型准备和控制系统的研究 [D]. 武汉：武汉理工大学，2006.

[7] 徐 巧，丛洪莲，张爱军，等. 纺织针织物 CAD 设计模型的建立与实现 [J]. 纺织学报，2014，35 (3)：136.

[8] 王 霞，蒋高明，丛洪莲，等. 基于互联网的纺织针织物计算机辅助设计系统 [J]. 北京：中国纺织工程学会，2017，38 (8)：150－155.

[9] BERNASCONI A，BODEI C，PAGLI L. Knitting for fun：a recursive sweater [M]. Berlin：Springer，2007.

[10] BERNASCONI A. On formal descriptions for knitting recursive patterns [J]. Journal of Mathematics & the Arts，2008，2 (1)：9－27.

[11] ZHAVIERA S E I，BOZOV S O. CAD module for knitting patterns design [C] //New York：ACM，2011：298－304.

[12] 赏仲天. 电脑横机的花型准备系统 [D]. 杭州：浙江大学，2005.

[13] 肖仕丽. 针织提花袜品计算机辅助快速分析设计系统的研究 [D]. 上海：东华大学，2007.

[14] 朱 艳. 针织机计算机辅助花样制作系统的研究 [D]. 杭州：浙江大学，2002.

[15] MIYAZAKI T，SHIMAJIRI Y，YAMADA M，et al. A knitting pattern recognition and stitch symbol generating system for knit designing [J]. Oxford：Elsevier，1995，29 (1)：669－673.

[16] 郭 艳. 电脑横机花型准备系统的设计与实现 [D]. 武汉：武汉理工大学，2006.

[17] 王元瑾，龙海如. 计算机辅助羊毛衫设计 [J]. 纺织导报，2001 (1)：33－35.

[18] MA Y. An integrated model for shaped knit garment design and development [D]. Raleigh：North Carolina State University，2013.

[19] 蔡立挺，傅建中，姚鑫骅. 嵌入式电脑横机花型数据的编译处理 [J]. 纺织学报，2008，29 (6)：113－116.

[20] 郭 思，蒋高明. 纺织提花毛圈织物计算机辅助设计 [J]. 北京：中国纺织工程学会，2014 (4)：142－147.

(上接第 121 页)

[11] 葛泉波，李 凯，张兴国. 基于多关键点检测加权融合的无人机相对位姿估计算法 [J]. 自动化学报，2024，50 (7)：1402－1416.

[12] 丘洪键，梁海平，卢耀安，等. 基于光学定位系统反馈的机器人末端位姿误差在线补偿方法 [J]. 工具技术，2024，58 (12)：136－141.

[13] 王 敏，孙景健，丁基恒，等. 基于 D-H 参数与拉格朗日联立方程的仿生水蛇机器人运动学分析及动力学建模 [J]. 机械工程学报，2024，60 (15)：134－148.

[14] 丁亚琼，贾寒光，万 凯. 基于自适应机器学习的视觉机械臂控制方法 [J]. 华南师范大学学报 (自然科学版)，2024，56 (3)：50－57.

[15] 张家维，白成超，郭继峰. 考虑基座振动抑制的双空间机械臂协同阻抗控制 [J]. 宇航学报，2024，45 (7)：1111－1122.

[16] 双 丰，刘旭兀，李少东，等. 基于 GPF-RRT＊的机械臂自主运动规划 [J]. 计算机集成制造系统，2023，29 (4)：1174－1185.

[17] 赵 雄，曹功豪，张鹏飞，等. 三自由度苹果采摘机械臂动力学分析与轻量化设计 [J]. 农业机械学报，2023，54 (7)：88－98.

[18] 陈洪月，蔡明航，杨辛未，等. 更换电铲钢丝绳专用机械臂架的结构及动力学分析 [J]. 工程设计学报，2023，30 (5)：590－600.

[19] 罗煜权，刘明鹏，陈姝文，等. 基于深度神经网络的充电负荷预测及优化调度模型 [J]. 电子设计工程，2025，33 (8)：92－95.

[20] 刘竟飞，姜 潮，倪冰雨，等. 基于主动学习与贝叶斯深度神经网络的高维多输出不确定性传播方法 [J]. 中国机械工程，2024，35 (5)：792－801.

[21] 刘 乐，孟德宇，常刘杰，等. 考虑时变非对称输出约束的机械臂固定时间自适应优化控制 [J]. 控制与决策，2025，40 (3)：833－842.

[22] 丁 卫，郑 云，钟宋义，等. 基于循环神经网络的 2-DOF 软体机械臂运动建模与控制 [J]. 上海大学学报 (自然科学版)，2024，30 (3)：522－531.

[23] 申 珅，曾建潮，秦品乐. 基于奖励与策略双优化的机械臂控制算法 [J]. 中北大学学报 (自然科学版)，2023，44 (6)：616－623.

[24] 邝逸灵，吴 迪，后国炜，等. 一种用于冗余机械臂解析逆运动学的多目标优化方法 [J]. 机械科学与技术，2024，43 (6)：934－942.

[25] 郑见云，史耕金，王 佑，等. 基于前馈增益调度的火电机组协调控制及应用 [J]. 计算机仿真，2023，40 (11)：96－100.