지능적 생산관리를 위한 온톨로지 기반의 상황인지 모형

류재환, 손종수, 정인정

고려대학교 컴퓨터정보학과
충남 연기군 조치원읍 서창리
Tel: +82-2-041-860-1342, Fax: +82-2-041-864-0014, E-mail:{berserkjan, mis026, chung}@korea.ac.kr

요약

RFID(Radio Frequency IDentification)는 RFID 리더를 통해 RFID 태그에 비접촉 방식으로 데이터를 읽고 쓸 수 있는 무선 인식 기술로서, 사용의 편리함 및 태그의 데이터 저장능력 등과 같은 장점으로 인해 많은 산업 현장에 적용되어 다양한 용도로 활용되고 있다. 이러한 가운데 ERP(Enterprise Resource Planning)와 같은 기업 관리 시스템과 연계한 RFID 시스템도 등장하고 있는데, 이와 같은 연계 시스템은 효과적인 물류 관리를 가능하게 할 뿐만 아니라, 기존 시스템에 실시간성을 부여함으로써 여러 가지 시간 관련 정보를 얻을 수 있도록 한다. 그러나 RFID 시스템은 능동적인 상황인지를 하지 못하기 때문에 RFID 시스템만으로는 지능적인 생산관리를 하기가 어렵다.

본 논문에서는 지능적인 생산관리를 위한 온톨로지 기반의 상황인지 모델을 제안하고, 이를 RFID 생산관리 시스템에 적용시켰다. 제안한 모델과 시스템을 통해 현재까지의 공정진행 상황 및 각 작업장의 생산 조건들을 파악하여 앞으로 진행될 공정을 안내함으로써 동적인 상황변화에 대한 적응력을 높였다. 또한, 생산현장에서 공정 진행에 관한 의사결정을 도움으로써 더 신속하고 효율적인 생산업무가 가능해졌다.

키워드:

RFID; ERP; 온톨로지; 상황인지; 생산관리;

1. 서론

ERP(Enterprise Resource Planning)는 기업의 인적, 물적 자원을 효율적으로 관리하기 위하여 개발된 전사적 자원관리 프로그램으로서 오늘날 그 중요성이 부각되어 많은 기업들이 사용하고 있다[5]. 또한, RFID(Radio Frequency IDentification) 역시 비접촉성, 다수 태그 동시 읽기, 태그의 데이터 저장능력 등의 장점으로 그 편리함과 효율성을 인정받아 여러 분야의 산업 현장 곳곳에 적용되었고, 그 추세는 아직도 계속되고 있다[4]. 이러한 가운데, 기존에 사용중인 ERP에 RFID를 연계한 시스템도 개발되고 있다. ERP와 연계한 RFID시스템은 여러 데이터들이 RFID 시스템을 통해 처리되는 즉시 ERP에도 반영함으로써 실시간성을 확보하는 장점을 갖는다[1][2].

한편, RFID 생산관리 시스템이 생산과정에서 발생하는 데이터를 관리하는 것에만 그치지 않고 복잡한 생산공정의 진행을 돕기 위해서는 생산과정을 지능적으로 관리할 수 있어야 하고, 이를 위해서는 수시로 변화하는 생산현장의 상황을 인지하고 처리할 수 있는 상황인지 모델이 필요하다. 우리는 기존 연구에서 워크플로우 상황인지 모델인 공정트리(Procedure Tree)[1][2]를 제안했고, 이를 사용하여 연구를 진행해왔다. 공정트리는 현재 공정의 진행상황 파악 및 앞으로 진행 가능한 공정의 제시 및 해당 공정으로의 유도에 성공하였으나 다음과 같은 문제점도 지니고 있었다.

- 이기종 간의 호환성이 떨어짐
- 작업장의 상황을 인지하여 생산가능 여부를 가늠하지 못하므로 효율적인 공정 안내가 불가능함

온톨로지(Ontology)는 어떤 관심 분야를 개념화하기 위해 명시적으로 정형화한 명세서로 정의되며, 대표적인 장점으로 이기종 분산처리 환경 하에서의 호환성과 관계 추론 능력을 들 수 있다[13]. 따라서, 본 논문에서는 위의 문제점을 보완하기 위한 방법으로써 공정트리를 활용하여 온톨로지 기반의 상황인지 모델로 발전시키고, 이를 ERP와 연계한 RFID 생산관리 시스템에 적용한다. 발전된 모델은 이기종 분산처리 환경하에서의 호환성을 확보하였고, 각 작업장의 생산 가능 여부를 판단하여 작업자의 의사결정을 도울 수 있다.

2절에서는 본 연구의 배경 지식으로 ERP와 연계한 RFID 생산관리 시스템 및 공정트리에 대하여 알아보고, 3절에서는 온톨로지 기반의 상황인지 모델을 제안하며, 이를 이용한 생산 현장의 상황인지 및 지능적 생산관리 과정은 4절에서

보인다. 끝으로, 5절에서는 결론 및 향후 과제를 제시한다.

2. 관련 연구 및 배경 지식

2.1 ERP와 연계한 RFID 생산관리 시스템

ERP는 인적, 물적 자원을 관리하는데 대단히 강력한 기능을 제공하지만, 자료를 입력하는 것은 사람이 하는 일이기 때문에 모든 자료를 실시간으로 입력하는 것은 매우 힘들다. 따라서, 실제로 변동된 데이터가 ERP에 입력되기까지는 다소의 시간차가 발생한다. ERP와 연계한 RFID 시스템은 현장에서 발생하는 모든 물적 자원의 변동 사항을 발생 즉시 처리하여 실시간성을 보장함으로써 이러한 문제를 해결할 수 있다. 따라서, RFID 시스템에서 처리된 데이터가 ERP에 자동으로 입력되는 시스템이 구축되면 언제라도 ERP를 통해서 모든 자원에 대한 데이터를 시간차 없이 확인할 수 있다[1].

실시간성이 보장되는 ERP는 다음의 항목과 같은 시간관련 정보를 제공한다.

- 품목별 현재 재고 현황
- 작업장별 현재 생산 완료량
- 각 작업장, 각 품목의 시간대별 생산량
- 단위 시간 당 평균 생산율
- 예상 작업 완료 시간

이와 같은 정보들은 정확한 현재 상황의 파악뿐만 아니라 앞으로의 상황 예측을 가능하게 함으로써 작업 속도에 유연성을 부여하고, 계획적인 생산이 이루어지도록 한다.

2.2 공정트리 (Procedure Tree)

공정트리는 생산관리 시스템이 현재 공정 진행 상황을 파악하고, 앞으로 진행 가능한 공정을 예측하기 위한 워크플로우 상황인지 모델이다. 모든 품목에 대한 생산 공정을 트리의 노드로 표현하고, 루트노드로부터 단말노드로 가는 다양한 경로로써 공정의 흐름을 나타낸다. 공정트리는 패스트리(PathTree)[5]를 수정하고 보완하여 만들어졌다. 패스트리란, Ziawei Han이 제안한 RFID 흐름 분석 기법인 RFID Cuboid에서 사용하는 자료구조이다. 저자는 상품이 이동할 때 특정한 단위의 묶음으로 이동되기 때문에 그룹별 관리가 필요하다는 생각으로 제안하였다. 트리에서 상위 노드와 하위 노드 사이의 이동 흐름 관계를 나타내는 GID(Generalized ID)를 사용하여 그룹화된 상품을 관리하는 방법으로 데이터를 일반화하였고, 이를 통해 데이터의 압축과 질의 처리의 효율성을 검증하였다. 그러나 다음과 같은 문제점으로 인해 패스트리를 RFID 생산관리 시스템에 그대로 적용하기에는 어려움이 있었다.

- 최소 관리 단위가 그룹 단위이기 때문에 상품의 개별적인 정보를 관리하기가 어려움
- 규모가 작은 기업의 경우에는 개별 단위 이동이 빈번하기 때문에 비효율적임
- 상품의 생산 과정에서 이전 공정의 추적, 현재 공정의 판단, 다음 공정의 예측과 같은 공정 관리를 하기에는 부적합함
- 상품의 흐름 상태 추적이 어렵기 때문에 공정 과정 중의 오류 탐지나 이동 동향 분석이 어려움

이러한 문제를 극복하기 위해 패스트리를 수정 및 보완하여 공정트리로 명명하였다. 공정트리는 패스트리와 비교하여 다음과 같은 차이점을 지닌다.

- 공정트리에서는 그룹화된 상품이 아닌 개별 상품에 대하여 GID를 할당함으로써 개별 상품에 대한 관리가 가능함
- 상품의 생산 계획을 바탕으로 최종 공정의 GID를 별도로 관리함으로써 공정의 추적, 예측과 같은 공정 관리가 가능함
- 패스트리에서는 소요된 시간을 time_in과 time_out 속성을 사용하여 표현하는 반면, 공정트리에서는 각 노드의 크기로 표현하여 별도의 계산 없이 각 공정별로 소요된 시간이나 가장 오랜 시간이 소요된 공정을 공정트리를 통해 바로 찾을 수 있음

3. 온톨로지 기반 상황인지 모델

3.1 공정트리를 기반으로 한 XML의 생성

공정트리는 노드와 에지의 집합으로 구성되어 있으며, 각 노드의 값은 공정코드와 해당 공정이 진행되는 작업장 코드의 쌍인 “WP_CD”로 표시된다. 예를 들어, W02 작업장에서 진행되는 10번 공정은 (10,W02)의 순서쌍으로 나타내고, 시스템에서는 작업장 코드의 W를 생략하여 “10.02”와 같이 간략화된 형태로 나타낸다. 각 순서쌍은 각 품목의 생산공정 내에서 유일한 값으로서 노드 n_i 의 값이 되고, 더 진행될 공정이 없는 말단 노드는 WP_CD 대신 “END” 값을 갖는다. 또한 에지 e_i 는 노드 n_i 로부터 노드 n_j 까지의 노드 사이에 방향성 있는 화살표로 표현되며, 이 화살표의 이름을 PID(Procedure ID)라고 한다. PID는 해당 에지의 아래에 연결된 노드의 인덱스로서 품목의 현재 상태를 나타내며, 각 PID는 부모 노드의 PID에 자식 노드의 순번(Sequence No.)을 붙임으로써 결정된다. 예를 들어, PID 가 12인 부모 노드가 3개의 자식

노드를 가지고 있는 경우, 자식 노드의 PID는 각각 12.0, 12.1, 12.2 가 된다. 그림 1은 이렇게 만들어진 공정트리의 일부를 보여준다.

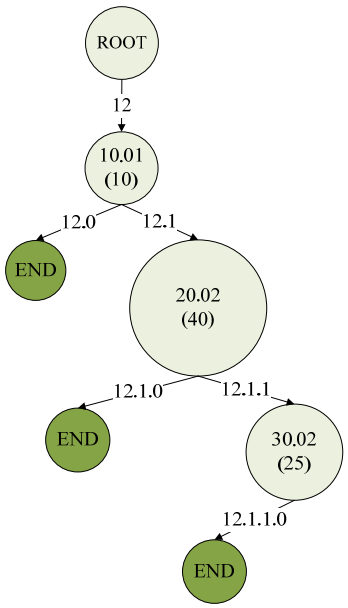

그림 1 – 공정트리의 구성 예

작업지시서에 부착된 RFID 태그에 저장된 데이터에는 생산중인 품목의 현재 상태를 나타내는 PID 값이 포함되어 있다. RFID 리더로 읽은 PID 값을 공정트리에서 탐색하여 해당 품목의 현재 공정과 다음 진행 가능 공정을 파악한다.

공정트리에서 루트 노드를 제외한 모든 노드들은 부모노드의 자식노드로 볼 수 있다. 따라서, 공정트리를 XML로 나타내면 자식노드들의 계층 구조로 나타낼 수 있다. 또한, 모든 노드들은 WP_CD를 나타내는 *name*, PID를 나타내는 *relation*, 단말 노드의 여부를 나타내는 *end*, 그리고 공정 소요시간을 나타내는 *cost*의 4가지 속성을 갖는다. 공정트리는 알고리즘 1,2를 통해 XML로 변환되며, 모든 변수는 이탤릭체로 표기한다.

알고리즘 1은 파라미터로 주어진 노드를 ROOT 노드로 하여 모든 노드를 순회한 결과를 XML파일에 기록한다. 또한 ROOT 노드의 속성 값으로 지정할 항목들을 파라미터로 하여 알고리즘 2를 호출한다. 알고리즘 2는 모든 자식노드들의 속성값을 부여하는 역할을 수행하며, 더 이상 남은 자식노드가 없을 때까지 재귀수행한다. 알고리즘 2의 setAttribute 메서드는 노드의 각 속성값을 지정하는 역할을 한다.

Traversal (*rootNode*)
1 *PT* ← Create Element (“ROOT”)
2 showGIDs (*rootNode*, “12”, *PT*)

알고리즘 1 – 공정트리의 순회

showGIDs (*childNode*, *PID*, *rootNode*)
1 **if** (*childNode* ≠ *rootNode*)
2 *childXMLNode* ←
Create Element (“CHILDNODE”)
3 setAttribute(“name”, name of *childXMLNode*)
4 setAttribute(“relation”, *PID*)
5 setAttribute(“cost”, consumed time)
6 **if** (*childNode* ≠ End Node) setAttribute(“end”, 0)
7 **else** setAttribute(“end”, 1)
8 *rootNode* ← *childXMLNode*
9 **for** *i*=0 to number of *childNode*
10 showGIDs (*i*-th child node of *childNode*,
PID + “.” + *i*, *rootNode*)

알고리즘 2 – 자식노드 생성 및 속성값 설정

그림 2는 알고리즘 1,2를 통해 XML로 변환된 공정트리를 보여준다.

```
<?xml version="1.0" encoding="utf-8" ?>
- <ROOT>
  - <CHILDNODE name="10.01" relation="12" end="0" cost="10">
      <CHILDNODE name="END" relation="12.0" end="1" cost="*" />
    - <CHILDNODE name="20.02" relation="12.1" end="0" cost="40">
        <CHILDNODE name="END" relation="12.1.0" end="1" cost="*" />
      - <CHILDNODE name="30.02" relation="12.1.1" end="0" cost="25">
          <CHILDNODE name="END" relation="12.1.1.0" end="1" cost="*" />
        </CHILDNODE>
      </CHILDNODE>
    </CHILDNODE>
  </ROOT>
```

그림 2 – XML로 기술된 공정 트리

3.2 XML의 온톨로지 변환

XML은 구조화된 데이터를 나타내기에 적합하기 때문에 공정트리를 잘 나타낼 수 있다[6]. 그러나 XML의 가장 큰 단점은 그저 문법을 잘 묘사할 뿐, 특정 도메인에서 의미를 인식할 수 있는 방법이 없다는 것이다. 이는 XML의 목적이 문서의 구조를 잘 나타내는데 있기 때문이다[7]. 온톨로지는 구조뿐만 아니라, 객체가 지닌 속성, 다른 객체 및 속성과의 관계표현에 적합하기 때문에 XML이 지닌 단점을 보완할 수 있다[8]. 따라서 본 논문에서는 컴퓨터가 노드 및 데이터 사이의 관계를 이해하여 처리할 수 있도록 하기 위해, XML을 다시 온톨로지로 변환한다. 온톨로지 언어로는 RDF(Resource Description Framework)[9]를 사용한다.

공정트리는 공정의 흐름을 나타낸 것이기 때문에 같은 공정이라도 흐름에 따라 여러 번 나타날 수 있다. 반면에 온톨로지에서는 인스턴스 ID의 중복을 허용하지 않는다. 따라서, XML로 기술된 공정트리를 온톨로지로 변환할 때는 같은 공정을 나타내는 인스턴스라 하더라도 ID를 모두 다르게 부여해야 한다. 본 논문에서는 각 인스턴스의 ID를 구분하기 위해 각 노드의 이름 뒤에 순차적으로 번호를 부여하였다. 그림 3은 공정트리를 기초로 생성된 XML을 온톨로지로 변환한 코드를 보여준다.

```
<?xml version='1.0' encoding='UTF-8'?>

<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'
  xmlns:Node='http://iis.korea.ac.kr/proceduretree/rdf#'>

  <rdfs:Class rdf:ID="Node">
    <rdfs:subClassOf rdf:about="#Node"/>
  </rdfs:Class>

  <rdf:Property rdf:ID="name">
    <rdfs:domain rdf:resource="#Node"/>
    <rdfs:range rdf:resource="&rdfs;Literal"/>
  </rdf:Property>

  <rdf:Property rdf:ID="Relation">
    <rdfs:domain rdf:resource="#Node"/>
    <rdfs:range rdf:resource="&rdfs;Literal"/>
  </rdf:Property>

  <rdf:Property rdf:ID="end">
    <rdfs:domain rdf:resource="#Node"/>
    <rdfs:range rdf:resource="&rdfs;Literal"/>
  </rdf:Property>

  <rdf:Property rdf:ID="cost">
    <rdfs:domain rdf:resource="#Node"/>
    <rdfs:range rdf:resource="&rdfs;Literal"/>
  </rdf:Property>

  <rdf:Description rdf:about="http://iis.korea.ac.kr/proceduretree/rdf/ROOT">
    <Node:name>ROOT</Node:name>
    <Node:relation>12</Node:relation>
    <Node:end>0</Node:end>
    <Node:cost>0</Node:cost>
    <Node:parent>Null</Node:parent>
  </rdf:Description>

  <rdf:Description rdf:about="http://iis.korea.ac.kr/proceduretree/rdf/10.01">
    <Node:name>10.01</Node:name>
    <Node:relation>12.0</Node:relation>
    <Node:end>0</Node:end>
    <Node:cost>0</Node:cost>
    <Node:parent>12</Node:parent>
  </rdf:Description>
```

그림 3 – 온톨로지로 변환된 공정트리의 일부

4. 온톨로지 기반의 상황인지 모형을 이용한 지능적 생산 관리

본 논문에서는 ㈜한국전산홈(Korea Computer Form, 이하 KCF)을 배경으로 한 예제를 보인다. KCF는 각종 고지서, 택배운송장, 안내문 등의 전산용지를 생산하는 기업으로, 산학협력 프로젝트를 통해 ERP와 연계한 RFID 생산관리 시스템이 적용되었다.

4.1 생산 작업장 상황인지

KCF의 생산공장에는 6개의 작업장이 존재하며 ERP 내에서는 각 작업장을 W01부터 W06까지의 코드로써 구분한다. 각 제품은 몇 단계의 공정순서를 거쳐 완제품으로 생산되는데, 공정순서 코드들 중 10번대 코드들은 모두 인쇄 공정으로서 주로 W01 작업장에서 진행되고, 나머지 코드들은 모두 가공 공정으로서 작업의 종류에 따라 W02부터 W06까지의 각 작업장에서 진행된다. 각 작업장에는 생산품목별로 여러 장의 작업지시서가 존재하며, 모든 작업지시서에는 해당 품목의 생산 정보를 담은 RFID 태그가 부착되어있다. 매 공정순서가 완료될 때 마다 이동형 RFID 리더로 작업지시서에 부착된 태그를 읽어서 해당 공정순서의 완료처리를 하며, 이 때 RFID 생산관리 시스템은 온톨로지로부터 다음에 진행 될 공정순서 정보를 받아 작업자에게 알려준다. 이러한 과정은 모든 작업장에서 동시 다발적으로 이루어지며, 모든 공정 진행 기록은 ERP 및 RFID 데이터베이스에 실시간으로 입력되어 온톨로지와 함께 생산 현장의 상황을 파악하는데 이용된다.

4.2 지능적 생산관리

W01 작업장에서 인쇄공정을 모두 마친 품목은 W02 작업장으로 이동하여 가공 공정을 거치게 되는데, W02 작업장의 작업 상태(유휴/생산중)에 따라서 W01 작업장에서의 공정을 마친 품목이 W02 작업장으로 이동할 것인지, 이동하지 않고 대기할 것인지 결정하게 된다. 이 때, 온톨로지 기반의 상황인지 모델을 이용하면 W02 작업장의 작업 상황을 미리 알아내어 W01 작업장의 실무자가 생산 완료 품목의 이동 여부에 대한 의사 결정을 지원할 수 있다. W02 작업장의 상황을 파악하기 위한 단계는 다음과 같다.

단계 1. RFID 데이터베이스로부터 W02 작업장에서 가장 최근에 이동형 RFID 리더를 통해 처리된 공정순서, 품목코드, 작업지시번호를 검색

단계 2. ERP에 저장된 생산 진행 내역으로부터 각 단계별 공정순서 및 작업장 코드를 검색

단계 3. 진행 순서대로 공정트리 온톨로지에서 노드를 추적하여 최종 PID를 도출

단계 4. 도출된 PID를 갖는 노드가 “END”이면 작업장은 유휴상태이고, 다음 노드가 “END”가 아니면 작업장은 가동중 상태

예를 들어, 품목코드가 ‘1B7C40043’인 제품의 경우, 4가지의 진행 흐름이 존재하며 주문자의 요청에 따라 미리 예정된 생산 계획에 따라 하나의 흐름을 선택하여 진행한다. 그 순서는 다음과 같으며, 본 논문에서는 3번의 경우를 선택하여 예제를 보인다.

1. ROOT → 10.01 → 20.02 → 30.02 → 40.02 → END
2. ROOT → 10.01 → 11.01 → 20.02 → 30.02 → END
3. ROOT → 10.01 → 11.01 → 20.02 → 30.02 → 40.02 → END
4. ROOT → 10.01 → 11.01 → 30.02 → 40.02 → 50.02 → END

공정 진행의 분기점에서는 그림 4와 같은 안내 화면이 나타나고, 작업자는 예정된 생산 계획에 따라 다음 공정인 20.02를 선택한다. 20.02에서 02는 W02 작업장을 의미하므로, RFID 시스템은 선택한 공정이 W02 작업장에서 진행되는 공정임을 확인하고 W02 작업장의 생산 상황 파악을 시작한다.

그림 4 – 공정 진행 분기 화면

1단계 작업으로, RFID 데이터베이스의 'PRF020T' 테이블에서 가장 최근 공정처리 정보를 가지고 온다. 'PRF020T' 테이블은 RFID 태그 정보가 변경된 기록을 저장하는 테이블로서, 다음과 같은 스키마로 구성되어 있다.

- ITEM_TCD – 품목에 대한 태그 코드
- ITEM_STATUS – 품목의 상태
- WK_ORD_NO – 작업지시번호
- WK_SHOP_CD – 작업장코드
- ROUT_SEQ – 공정순서 코드
- WR_DT – 기록일시

W02 작업장에서 가장 최근에 처리된 공정정보는 WK_SHOP_CD가 'W02'이고 WR_DT 값이 가장 큰 행을 검색하여 얻을 수 있다. SQL쿼리문과 그 결과는 그림 5와 같다.

```
Select  *
From    PRF020T
Where   WK_SHOP_CD = 'W02' and
        WR_DT = ( Select max(WR_DT)
                  From   PRF020T )
```

	ITEM_TCD	SEQ_CD	VAR_QTY	EMP_PRSN	WR_DT	ITEM_STATUS	WK_ORD_NO	WK_SHOP_CD	ROUT_SEQ
1	30	1	40	PKH	2009-10-09 11:35:17.043	12.0.30.3.1.1	2008040112	W02	40

그림 5 – 가장 최근에 처리된 공정정보

2단계 작업으로, 1단계에서 얻은 작업지시번호와 품목코드를 통해 ERP 데이터베이스에서 해당 작업의 진행기록을 검색한다. SQL쿼리문은 ERP가 생성하는 SQL쿼리문을 캡쳐하여, 검색 목적에 맞도록 조건절을 수정하여 작성하였다. 질의의 결과로 현재까지 수행된 공정순서 코드들을 얻는다. 그림 6은 질의의 결과로 얻은 데이터이고, 공정진행순서는 ROUT_SEQ 값의 오름차순이다.

	ORG_CD	WK_ORD_NO	ROUT_SEQ	ITEM_CD	ITEM_NM
1	12	2008040112	10	1B7C40043	계약전알릴의무사항II - LTC상품
2	12	2008040112	11	1B7C40043	계약전알릴의무사항II - LTC상품
3	12	2008040112	20	1B7C40043	계약전알릴의무사항II - LTC상품
4	12	2008040112	30	1B7C40043	계약전알릴의무사항II - LTC상품
5	12	2008040112	40	1B7C40043	계약전알릴의무사항II - LTC상품

그림 6 - 2단계 질의로 얻은 공정 순서 진행 기록

3단계에서는 2단계에서 얻은 공정순서 코드들을 온톨로지에서 추적한다. 온톨로지 검색에는 SPARQL[10]을 이용하였으며, 검색엔진은 Jena[11] 2.6.2 버전을 사용하였다.

추적은 ROOT 노드를 부모로 갖는 자식노드를 찾는 것을 시작으로 한다. ROOT 노드의 PID는 12이므로 *parent*값이 12인 노드를 찾으면 *relation*이 12.0인 10.01노드가 검색된다. 다시 *parent*값이 12.0인 노드를 찾으면 *relation*이 12.0.30인 11.01과, *relation*이 12.0.1인 20.02가 검색된다. 2단계 작업의 결과로 얻은 공정 진행 순서를 보면, 10번공정 다음에 진행된 공정은 11번공정이므로 11.01을 선택한다. 이와 같은 방식으로 40번 공정까지 추적을 계속한다. 그림 7은 부모노드의 PID가 12인 자식노드를 찾는 SPARQL 쿼리문이며, 결과로 도출된 PID는 다시 쿼리문의 조건절로써 검색에 사용한다. 그림 8은 온톨로지에서 공정 진행 순서를 추적한 결과이다.

```
PREFIX Node:<http://iis.korea.ac.kr/proceduretree/rdf#>

SELECT ?name ?relation
WHERE
  { ?x  Node:name  ?name .
    ?x  Node:relation  ?relation .
    ?x  Node:parent  "12" .
  }
```

그림 7 – ROOT 노드를 찾는 SPARQL 쿼리문

마지막으로 4단계에서는 3단계에서 얻은 PID를 통해 자식 노드들을 검색한다. 검색결과, 그림 8의 6번의 결과를 얻었고, 유일한 자식노드가 "END" 이므로 W02 작업장은 40번 공정을 마지막으로 처리한 후 현재 유휴상태이다. 도출된 결과는 작업자에게 보고되고, W01 작업장에서의 공정을 마친 품목은 W02 작업장으로의 이동이 결정된다.

```
1. parent="12"
----------------------
| name    | relation |
======================
| "10.01" | "12.0"   |
----------------------

2. parent="12.0"
-----------------------
| name    | relation  |
=======================
| "11.01" | "12.0.30" |
| "20.02" | "12.0.1"  |
-----------------------

3. parent="12.0.30"
--------------------------
| name    | relation     |
==========================
| "30.02" | "12.0.30.4"  |
| "20.02" | "12.0.30.3"  |
--------------------------

4. parent="12.0.30.3"
----------------------------
| name    | relation       |
============================
| "30.02" | "12.0.30.3.1"  |
----------------------------

5. parent="12.0.30.3.1"
-----------------------------
| name    | relation        |
=============================
| "40.02" | "12.0.30.3.1.1" |
| "END"   | "12.0.30.3.1.0" |
-----------------------------

6. parent="12.0.30.3.1.1"
-------------------------------
| name    | relation          |
===============================
| "END" | "12.0.30.3.1.1.0" |
-------------------------------
```

그림 8 – 공정 진행순서 추적 결과

5. 결론 및 향후 과제

본 논문에서는 공정트리를 수정 및 보완하여 온톨로지 기반의 지능형 상황인지 모델로 발전시키고, 이를 ERP와 연계한 RFID 생산관리 시스템에 적용하였다. 발전된 상황인지 모델은 온톨로지를 기반으로 하여 이기종 호환성을 확보하였다. 또한, 제안한 모델을 통해 다음 공정을 제시하는 것뿐만 아니라 공정의 진행 가능 여부에 따른 생산 작업의 투입, 혹은 대기의 의사결정을 돕는 역할까지 할 수 있게 되었다. 따라서, 본 논문에서 제안한 상황인지 모형을 현업에 적용하면 실무자들은 시스템의 도움을 받아 신속한 의사결정과 더 효율적인 작업 수행이 가능하다.

제안한 모델은 공정트리를 기초로 만들어졌기 때문에 현재까지는 공정의 흐름에 대한 부분에만 적용된 상태이며, 앞으로는 활용 범위를 더욱 넓히기 위해 제안한 모델과 함께 규칙언어를 사용하는 연구를 진행할 것이다. 규칙언어를 이용한 추론 과정을 통해 생산 관리의 더 많은 부분에서 더욱 복잡하고 어려운 의사결정을 지원할 수 있도록 연구하고, 웹 서비스를 연계하여 소비자에게 주문한 상품의 생산 진행 현황을 보여줄 수 있도록 하는 방향으로의 연구도 유익할 것으로 보인다.

참고문헌

[1] 류재환, 권경락, 윤여창, 손종수, 정인정. (2009). “RFID 기반의 상황인지 생산관리 시스템”, *제31회 한국정보처리학회 춘계학술발표대회 논문집* 제16권 제1호 pp. 569-572.

[2] Kyunglag K., Yeochang Y., Jaehwan R., Jongsoo S., and Injeong C. (2008). “RFID Warehouse Management in the Small and Medium Enterprises based on Manufacturing Industry”, *The 3rd ICUT*, pp. 80-86.

[3] Jiawei H., Jaegil L., Hector G., and Xiaolei L. (2008). “Mining Massive RFID, Trajectory, and Traffic Data Sets”, *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining.*

[4] R Angeles. (2005). “RFID technologies: supply-chain applications and implementation issues”, *Information Systems Management.*

[5] Esteves, J., and Pastor, J. (2001). “Enterprise Resource Planning Systems Research: An Annotated Bibliography”, *Communications of AIS*, 7(8) pp. 2-54.

[6] Bray T., Paoli J., Sperberg-McQueen C. M., Maler E., and Yergeau E. (2008). “Extensible Markup Language (XML) 1.0 (Fifth Edition)”, *W3C Recommendation.*

[7] Derkers, S., Melnik S., Harmelen F., and Fensel D. (2000). “The Semantic Web: the roles of XML and RDF”, *Internet Computing*, IEEE Volume 4, Issue 5, pp.63 - 73.

[8] Berners-Lee T., Hendler J., and Lassila O. (2001). “The Semantic Web”, *Scientific American.*

[9] Klyne G., and Carroll J.J. (2004). “Resource Description Framework (RDF):Concepts and Abstract Syntax”, *W3C Recommendation.*

[10]Brickley D., and Guha R.V. (2004). “RDF Vocabulary Description Language 1.0: RDF Schema”, *W3C Recommendation.*

[11]Prud'hommeaux E., and Seaborne A. (2008). “SPARQL Query Language for RDF”, *W3C Recommendation.*

[12]http://www.openjena.org/

[13]Gruber TR. (1993). “A translation approach to portable ontology specifications”, *Knowledge acquisition*, 5(2):199-220.