

Using LEL and scenarios to derive mathematical programming models. Application in a fresh tomato packing problem

Alejandra Garrido^{a,b}, Leandro Antonelli^a, Jonathan Martin^a, M.M.E. Alemany^c, *Josefa Mula^c

^a LIFIA, Fac. de Informática, Univ. Nac. de La Plata, La Plata, Argentina, garrido@lifa.info.unlp.edu.ar,
lanto@lifa.info.unlp.edu.ar, jonathan.martin@lifa.info.unlp.edu.ar.

^b CONICET, Argentina

^c Research Centre on Production Management and Engineering (CIGIP), Universitat Politècnica de València, Valencia, Spain,
mareva@cigip.upv.es. *Corresponding author: fmula@cigip.upv.es

Abstract

Mathematical programming models are invaluable tools at decision making, assisting managers to uncover otherwise unattainable means to optimize their processes. However, the value they provide is only as good as their capacity to capture the process domain. This information can only be obtained from stakeholders, i.e., clients or users, who can hardly communicate the requirements clearly and completely. Besides, existing conceptual models of mathematical programming models are not standardized, nor is the process of deriving the mathematical programming model from the concept model, which remains ad hoc. In this paper, we propose an agile methodology to construct mathematical programming models based on two techniques from requirements engineering that have been proven effective at requirements elicitation: the language extended lexicon (LEL) and scenarios. Using the pair of LEL + scenarios allows to create a conceptual model that is clear and complete enough to derive a mathematical programming model that effectively captures the business domain. We also define an ontology to describe the pair LEL + scenarios, which has been implemented with a semantic mediawiki and allows the collaborative construction of the conceptual model and the semi-automatic derivation of mathematical programming model elements. The process is applied and validated in a known fresh tomato packing optimization problem. This proposal can be of high relevance for the development and implementation of mathematical programming models for optimizing agriculture and supply chain management related processes in order to fill the current gap between mathematical programming models in the theory and the practice.

Keywords: Language extended lexicon (LEL); scenarios; software engineering; mathematical programming; fresh tomato packing.

1. Introduction

There is an increasing interest in mathematical programming models for optimal decision support applications (Dominguez-Ballesteros et al., 2002). Indeed, the development of optimization and decision support tools is needed to obtain all the benefits of transactional information technology (IT), improving the economic performance and customer satisfaction of supply chains (Grossmann, 2005). Along these lines, mathematical programming models have been demonstrated to be powerful optimization tools to support decision makers in many supply chain processes such as: production planning (Alemany et al., 2013), order promising (Alemany et al., 2018; Grillo et al., 2017), shortage planning (Esteso et al., 2018), supply chain production and transport planning (Mula et al., 2010), among others. The agriculture sector also faces many

42 complex problems for optimization (Saranya & Amudha, 2017) as it has been reported in some
43 recent works (Cid-Garcia & Ibarra-Rojas, 2019; Grillo et al., 2017; Liu et al., 2019). Some
44 revisions about mathematical programming models applied to different problems in agriculture
45 can be found for supply chain design (Esteso, Alemany, & Ortiz, 2018), fresh fruit supply chain
46 management (Soto-Silva et al., 2016), agribusiness supply chain risk management (Behzadi et al.,
47 2018) and crop planning (Jain et al., 2018), among others.

48 Once formulated, mathematical programming models are often implemented as part of a decision
49 support system (DSS), which is thus called model-driven DSS. We refer readers to the work of
50 Udias et al. (2018) regarding an example of recent agricultural model-based DSS. These types of
51 DSSs allow the user to make what-if analysis and define different scenarios without the need to
52 understand the complexities of mathematical programming models (Mundi et al., 2013). Mir et al.
53 (2015) provide an extensive revision of DSS application in agriculture noting their main
54 weaknesses, most of them related to the poor involvement of stakeholders in the DSS construction
55 process, which our methodology aims to overcome: failure to support stakeholder participation
56 before and after development stages, failure to support the relationship between stakeholders and
57 experts/developers, low adaptation, complexity with user inputs and under-definition of end users.
58 Moreover, the construction of mathematical programming models is a complex and very time-
59 consuming process, which requires an expert to acquire a deep understanding of the modelling
60 domain, context of use, decision making activity, and to learn the complete set of constraints from
61 the problem under study. All this problem knowledge should be acquired before the model is
62 constructed, because any change that occurs afterwards might imply a whole redesign and
63 implementation of the model. For this reason, mathematical programming model construction
64 should be preceded by a conceptual modelling activity whose natural use in the field of applied
65 mathematics has been pointed out by Lesh (1981) and Lesh et al. (1983).

66 During conceptual modelling, different tools are used, like Bizagi, BPWin, iGrafx, Process
67 Modeler, System Architect and Visio, to help understand the domain, the flow of data, products,
68 decisions and the interaction among parties, and to elicit requirements as completely as necessary
69 (Armengol et al., 2015; Giannoccaro & Pontrandolfo, 2001; Hernández et al., 2008; Mula et al.,
70 2006; Pérez Perales et al., 2012). However, there has been no consensus or standardization in this
71 regard.

72 The process of creating a conceptual model may be closely compared with the process of
73 requirements elicitation for any software system. In particular, we claim that a very important
74 aspect of conceptual models is that they should allow their iterative, incremental and collaborative
75 construction. Indeed, agile methods in requirements engineering have demonstrated the
76 importance of managing the inherent complexity of a system specification in an incremental and
77 iterative manner (Schön et al., 2017). A relevant technique to elicit requirements and get a clear
78 and complete understanding of a domain is the use of scenarios (Leite et al., 2000). The description
79 of scenarios ranges from visual (storyboards) to narrative (structured text) (Young, 2004). They
80 are constructed iteratively on the basis of a universe of discourse (UofD), i.e., a domain's
81 vocabulary or lexicon. Leite and Franco (1993) named it the language extended lexicon (LEL). It
82 is a meta-language used to gather or elicit requirements, which aims at describing the meaning of
83 words and phrases specific to a given application domain. It has three convenient characteristics
84 in the context of analytical modelling: easy to learn (Cysneiros & Leite, 2001), easy to use (Gil et
85 al., 2000) and good expressiveness (Kaplan et al., 2000). Moreover, there exist specific rules to
86 derive LEL elements into scenario elements, and scenarios retrofit LEL's vocabulary in a very
87 incremental and iterative construction process (Leite et al., 2000). Here, we claim that the above
88 quality characteristics and the construction process of the pair LEL + scenarios make them an
89 adequate conceptual model from where mathematical programming models could be
90 systematically derived.

91 In this article, we propose a novel methodology to guide the derivation of mathematical
92 programming model elements from a conceptual model created with LEL + scenarios. Our
93 derivation proposal consists of several rules that map conceptual model elements (either LEL
94 vocabulary items or scenario elements previously derived from LEL) into mathematical
95 programming model elements. The added benefit of this methodology is that it provides
96 traceability from vocabulary and requirements specification to each mathematical programming
97 model element. This traceability becomes very important when a change is necessary in the
98 conceptual model, to know the particular place in the mathematical programming model
99 specification where the change will have an impact.

100 In order to give further support to the derivation process, we relate conceptual model creation to a
101 knowledge building process of collective creation between stakeholders and analysts. This process
102 emphasizes the production and continuous improvement of knowledge parts (Moskaliuk et al.,
103 2009), and it is usually supported with a web-based knowledge building community like a
104 mediawiki (Baraniuk et al., 2004). Thus, we have constructed a semantic mediawiki based on an
105 ontology for the collaborative creation of the conceptual model based on LEL and scenarios.
106 Moreover, the semantic mediawiki is used to semi-automatically derive mathematical
107 programming models.

108 Thus, the paper proposes a novel methodology that connects the areas of requirements engineering
109 and agile methods with conceptual modelling in order to create mathematical programming
110 models that capture the agriculture research domain more effectively and completely.
111 Summarizing, the main contributions of this paper are: (i) a proposal for utilizing the pair LEL +
112 scenarios to create conceptual models that gather the vocabulary of decision makers and specify
113 the domain knowledge necessary to build a mathematical programming model; (ii) a rule-based
114 methodology for the systematic derivation of mathematical programming models from the
115 conceptual model generated by LEL + scenarios; (iii) a knowledge model designed over an

116 ontology based on the description of LEL + scenarios; and (iv) an extensible tool consisting of a
117 semantic mediawiki that allows to partially automate the mathematical modelling derivation. To
118 the best of our knowledge, the methodology proposed in this article is the first approach to
119 standardize the development of a conceptual model and its consequent translation to a
120 mathematical programming model, and no other requirement elicitation method in the field of
121 software engineering has been adapted for the derivation of mathematical programming models.
122 Moreover, the derivation process provides traceability from requirements to mathematical
123 programming model elements to deal with changes more effectively.

124 The rest of the paper is organized as follows. Section 2 provides a literature review on
125 mathematical programming models, conceptual models, requirements elicitation, LEL and
126 scenarios and knowledge building. Section 3 describes our methodology for the systematic
127 derivation of mathematical programming model elements from the conceptual model generated
128 by LEL+ scenarios. Section 4 applies the method to derive a linear mathematical programming
129 model for fresh tomato packing. Section 5 defines an ontology for LEL + scenarios on which we
130 based the construction of a semantic mediawiki for the semi-automatic derivation of the
131 mathematical programming model. Section 6 provides conclusions and future research directions.

132 **2. Literature review**

133 **2.1 Mathematical programming models**

134 Mathematical programming involves finding the values of some variables that, subject to certain
135 constraints, maximize or minimize an objective function. We assume that deterministic
136 mathematical programming models have a generic structure: a definition part and a modelling part
137 (Pérez et al., 2010; Shapiro, 1993). Table 1 provides a description of mathematical programming
138 model parts.

139 **2.2 Conceptual mathematical modelling**

140 Siau (2004) defines conceptual modeling as the process of formally documenting a problem
 141 domain to achieve understanding and communication between the different participants.
 142 Developing conceptual models means specifying the essential objects, or components, of the
 143 system to be studied, and the relationships or types of exchanges between the objects that affect
 144 the functioning of the system (Lezoche et al., 2012). From the abstraction of conceptual models
 145 emerges the concept of reference models, which are generic conceptual models that formalize
 146 recommended practices for a given domain (Pesic & van der Aalst, 2005).

147 **Table 1.** Mathematical programming model parts.

MATHEMATICAL PROGRAMMING MODEL PART	ELEMENT	DESCRIPTION	EXAMPLES
Definition part	Indexes	Objects or concepts of the model. The number of elements of a class of objects provides the number of instances of this class	Machines (m) Products (i)
	Sets	Group of instances of one or several indexes that meet certain characteristics or constraints	Group of products that can be processed by each machine $P(m)$
	Parameters	Known characteristics of one or several elements (indexes) over which is not possible to act	Capacity of each machine (Cap_m) Production cost for each product on each machine (PC_{im})
	Decision variables	Unknown characteristics of one or several elements (indexes) over which it is possible to act (decision-maker can determine their value)	Quantity to be produced of each product on each machine every time period (X_{imt})
Modelling part	Objective function	Goal/s to be optimized (minimize or maximize)	Maximize profits Minimize costs Minimize time
	Constraints	Problem limitations that should be respected for every combination of the decision variables	Availability of resources (e.g. machines' capacity) Company policies (e.g. service level) Logic or implicit constraints (e.g. flow balance, positive quantities)
Decision-maker	Decision-maker	Person who makes the decision	Planner, manager
Temporal characteristics	Time-horizon	The length of time (with a beginning and end date) over which a problem is optimized	A year, six months, etc.
	Time-period	Space of time into which a time-horizon is divided	Seconds, minutes, hours, days, weeks, months or years
	Replanning time period	Space of time in which the plan is calculated again	Seconds, minutes, hours, days, weeks, months or years

148
 149 In the field of conceptual mathematical modelling, Schneeweiss (2003a) identifies different
 150 classes of distributed decision making (DDM) problems in supply chain management. The same
 151 author derives the coupling equations for the most usual cases in coordinating the supply chain
 152 (Schneeweiss, 2003b). However, the coupling equations are still of a very general, almost verbal
 153 character. In this context, Alemany et al. (2007) propose a reference mathematical programming

154 model for collaborative planning that addresses two of the challenges of DDM: the spatial and
155 temporal interdependencies. Alemany et al. (2011) developed an application to support the
156 integrated modelling and execution of the supply chain collaborative planning process made up of
157 several decisional centers which make decisions based on mathematical programming models.
158 However, the formulation of the own specific decisional characteristics of each decision center
159 (micro-decision view) mainly relies on the ability of the mathematical programming modeler.
160 Moreover, Pérez-Perales et al. (2012) propose a framework to support modelling the decisional
161 view of collaborative planning through mathematical programming models.

162 While the above studies are very useful, this article provides further tool-supported guidance and
163 a precise specification in terms of derivation rules to derive mathematical programming models
164 from a conceptual model, as opposed to general descriptions or relying in the modeler's ability.

165 **2.3 Process of mathematical programming model formulation**

166 Since conceptual models are not standardized, neither is the process of deriving the mathematical
167 programming model from the concept model, which remains ad hoc. In this sense, Raghunathan
168 (1996) proposed a methodology to design a DSS with its underlying mathematical programming
169 and data models. The methodology includes six steps: (i) problem domain analysis, (ii) database
170 design, (iii) modelbase design, (iv) database/modelbase integration design, (v) problem/decision
171 maker characteristics and (vi) specific DSS design. However, the setting of Raghunathan (1996)
172 is a classroom, so the problem statement is completely specified from the start. Alternatively, our
173 proposal is inspired in the current, agile way of system specification, which recognizes that the
174 construction of any model should be iterative and incremental. Additionally, the use of entity-
175 relationship modelling by Raghunathan (1996) has two implications: a) the problem must be
176 simple, otherwise the diagram is not even readable; and b) stakeholders may not be able to
177 understand it. Contrarily, we use a natural language and a semi-structured scenario specification

178 for the conceptual model, which is more appealing for a system specification that incorporates
179 stakeholders in the process.

180 Furthermore, Dominguez-Ballesteros et al. (2002) define different stages in the process of
181 deterministic and stochastic linear programming model formulation and implementation:
182 conceptualisation (data collection and study of the problem), data modelling (categorisation and
183 abstraction of the data), algebraic form (modeller's form), translation (matrix generator/modelling
184 language), machine-readable form (algorithm's form), solution and solution analysis. However,
185 the stages can be understood as guidelines for the mathematical programming modeler more than
186 a derivation procedure. That is, unlike Raghunathan (1996) and to the best of our knowledge, no
187 methodology or structured modeling language is proposed for the conceptualization stage.

188 **2.4 Requirements elicitation**

189
190 Requirements elicitation is the process that analysts follow to ensure a correct understanding of
191 stakeholders' needs and the domain specification before a system is designed and implemented
192 (Leite et al., 2000). In this regard, Geisser and Hildenbrand (2006) state that software requirements
193 are very complex and a multitude of stakeholders participate in their description (Geisser &
194 Hildenbrand, 2006). They propose a method called CoREA that covers collaborative requirements
195 elicitation in a distributed environment as well as quantitative decision support for distributed
196 requirements prioritization and selection. Our proposed approach also relies on collaborative
197 knowledge acquisition and description, with the added advantage of using this knowledge base as
198 a conceptual model from where a mathematical programming model can be derived through the
199 application of a set of rules.

200 Closer to our work, Laporti et al. (2009) propose an approach to develop system requirements in
201 an iterative and collaborative way. Experts in the domain collaborate to build narrative
202 descriptions of stories. Then, these stories are used as input to describe scenarios, which are in
203 turn used to define use cases. In this regard, our proposal is similar since it considers the collective

204 construction of LEL + scenarios and a mapping between LEL, scenarios and mathematical
205 programming models. The difference is that the output of the transformation in the work by Laporti
206 et al. (2009) is still a textual, semi-structured representation (use cases), which does not distance
207 much of the previous products, whereas in our case the output is a structured mathematical
208 programming model, so the mapping requires a more complex strategy that includes a precise
209 representation of the relations among model elements. Our approach uses two existing techniques
210 for requirements elicitation: the LEL (Leite & Franco, 1993) and scenarios (Leite et al., 2000).
211 The LEL is a very convenient tool for both stakeholders with no technical skills and analysts, since
212 it conforms to the mechanism used by the human brain to organize knowledge (Oliveira et al.,
213 2007), which makes it easy to learn while having good expressiveness. The process to build the
214 LEL is comprised of six steps (Breitman & Leite, 2003; Kaplan et al., 2000), which allow
215 constructing a list of terms classified in four categories (see Table 2).
216 Turning into scenarios, they can be used in different stages of software development, from
217 clarifying business processes and describing requirements to providing the basis of acceptance
218 tests (Alexander & Maiden, 2004). Leite et al. (2000) propose a template with six elements to
219 describe scenarios (see Table 2), which are derived from the LEL following a methodology
220 consisting of five steps: (i) to identify main and secondary actors, i.e., LEL symbols that belong
221 to the subject type; (ii) to identify scenarios within the behavioral responses of symbols chosen as
222 actors; (iii) to define the scenario goal based on the notion of the verb symbol in which the scenario
223 is based; (iv) to identify the scenario resources, searching in the notion of the verb that created the
224 scenario, for LEL symbols of the object category; and (v) to derive episodes from each behavioral
225 response of the verb that identified the scenario.
226

227 **Table 2.** LEL categories and scenarios elements.

Category	Characteristics	Notion	Behavioral responses
Subject	Active elements which perform actions	Characteristics or condition that subject satisfies	Actions that subject performs
Object	Passive elements on which subjects perform actions	Characteristics or attributes that object has	Actions that are performed on object
Verb	Actions that subjects perform on objects	Goal that verb pursues	Steps needed to complete the action
State	Situations in which subjects and objects can be located	Situation represented	Actions that must be performed to change into another state
Attribute	Description		
Title	Name that describes the scenario		
Goal	Conditions and restrictions to be reached after the execution of the scenario		
Context	Conditions and restrictions that are satisfied and constitute the starting point of the scenario execution.		
Actors	Agents that perform actions during the scenario starting from the context to reach the goal		
Resources	Products and elements used by the actors to perform actions		
Episodes	Steps executed by the actors using the resources starting at the context to reach the goal		

228
 229 **2.5 Ontologies and knowledge building**
 230
 231 Ontologies define the common vocabulary in which shared knowledge from a domain of discourse
 232 is represented (Gruber, 1993; 1995). They can be constructed in two ways, domain dependent and
 233 generic. CYC (Lenat, 1995), WordNet (G.A. Miller, 1995) and Sensus (Swartout et al., 1996) are
 234 examples of generic ontologies. A benefit of using a domain ontology is to attain the shared and
 235 agreed definition of a semantic model of domain data and the links between different types of
 236 semantic knowledge, which makes it suitable in formulating data searching strategies for
 237 information retrieval (Munir & Sheraz Anjum, 2018).
 238 Furthermore, a semantic mediawiki defined over an ontology provides a web-based support for a
 239 knowledge building community (Baraniuk et al., 2004). We have used a semantic mediawiki in
 240 this work to allow for the collaborative definition of LEL + scenarios of the problem domain and
 241 for the semi-automatic derivation of the mathematical programming model using the mediawiki's
 242 query engine.

243 **3. Methodology for mathematical programming model derivation**

244 **3.1. Conceptual model construction and methodology overview**

245 The mathematical programming model derivation process starts from an existing conceptual
246 model consisting of a complete or close to complete specification of LEL + scenarios of the
247 system. There are three variations that we propose to the original definition of LEL and scenarios
248 for the specific goal of generating mathematical programming models. The first is that we do not
249 use the terms in the “state” category of the LEL, because no mathematical programming model
250 element is derived from them. The second is that we distinguish attributes inside the notion of
251 symbols, especially those that become scenario’s actors and resources. That is, an actor is a LEL
252 subject, and as such it will have a notion with its conceptual definition. We call attributes to the
253 terms that characterize the actors and appear in their notion, usually after the verb “has”. Similarly,
254 a resource is a LEL object with a notion that names its attributes. In turn, attributes are also defined
255 as LEL objects, and this is the reason that attributes are underlined in the notion of the actor or
256 resource that they characterize, describing a relation between LEL terms. The third variation that
257 we propose is related to specifying the temporal location inside scenarios’ context element with
258 more detail, identifying three fields: time horizon, time period and replanning time period.

259 In order to provide a better understanding of the proposed methodology, Figure 1 depicts its two
260 main stages (Conceptual model construction and Mathematical programming model construction),
261 the phases in the construction process for each stage, and two different levels of iteration. There
262 is one iteration cycle that occurs often in the construction of the conceptual model, where scenarios
263 may retrofit the LEL, and a second level or global iteration cycle, between the conceptual model
264 and the mathematical programming model, which should not be as usual. The methodology
265 provides traceability by way of rules that specify the source of each mathematical programming
266 model element. Therefore, this methodology is robust enough to actually afford changes in the
267 mathematical programming model construction.

268 The input for the whole process, as Figure 1 shows, is the domain knowledge obtained from
269 stakeholders and documents. The first phase consists in the LEL specification, which is created by
270 a system analyst together with the stakeholders and, if possible, the expert in mathematical
271 modelling, thus creating a multi-disciplinary team. They should identify the sources of knowledge,
272 define the LEL, verify and validate it. The second phase consists in specifying scenarios using the
273 knowledge captured in the LEL. If during the description of the scenario, it is noticed that more
274 knowledge from the domain is needed, the process goes back to phase 1.

275 When the knowledge captured by the LEL + scenarios appears adequate and complete, the third
276 phase (Mathematical Programming Model Derivation) begins. This phase uses the knowledge
277 captured in LEL and scenarios to derive the mathematical model. The mathematical programming
278 derivation proposal consists of several rules that map conceptual model elements (either LEL
279 vocabulary items or scenario elements previously derived from LEL) into mathematical
280 programming model elements. The derivation process may be carried out manually by the
281 mathematical programming expert, possibly together with the system analyst. In addition, we
282 provide tool support for a semi-automatic derivation through a semantic mediawiki. Even with a
283 tool support, manual revision from the mathematical programming expert will be necessary, since
284 it is not possible to automatically create the equations that model constraints from a textual
285 description, although we can isolate the sentence that contains a constraint from the conceptual
286 model. In Fig. 1, the numbering of steps in the mathematical programming model derivation phase
287 denotes a sequence in which rules should be applied. Moreover, at the end of each of these steps,
288 the manual intervention of the mathematical programming expert is advised to prevent an overly
289 complex mathematical programming model (as it could happen with a large number of indexes)
290 or to spot missing items in the conceptual model. Furthermore, it could also be detected that more
291 knowledge from the LEL and Scenarios is needed, and in that case, the process goes back to the
292 conceptual model construction.

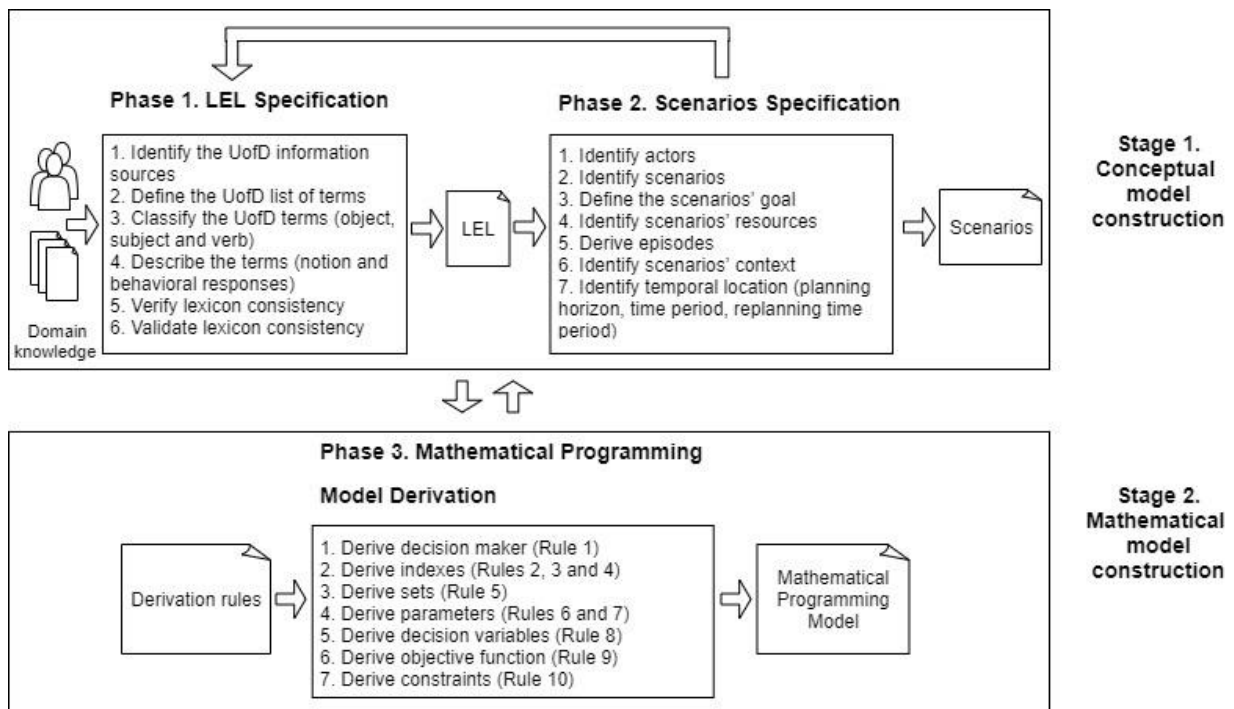


Fig. 1. Methodology for mathematical programming model derivation.

3.2. Mathematical programming model derivation

This section presents the rules that allow deriving a mathematical programming model from a conceptual model composed of LEL + scenarios. Below we present a detailed description of each derivation rule, listed by its rule number. Note that rule numbers do not dictate an order of application except for the order dictated by the methodology and outlined in Figure 1. Following this description, Table 3 provides a summary of the rules.

Rule 1. The main actor of the base scenario becomes the decision maker in the mathematical programming model

Main actors are those who execute actions in the **domain**, in this case, those making decisions. We consider the main actors to be a single person or several persons playing the same role, i.e., making a centralized decision. There will be a base scenario that derives from the behavioral response of the single main actor, who will be the decision maker in the mathematical programming model.

310 **Rule 2. The time period of the planning horizon defined in the temporal location, inside the**
311 **context of the base scenario, becomes an index of the mathematical programming model.**
312 **Moreover, other data objects related to time in the temporal location could also become**
313 **indexes.**

314 The base scenario should specify the temporal location in its context attribute. Particularly, the
315 time period specifies the regular intervals in the time horizon at which different decisions are to
316 be made. If such time period exists, Rule 2 is applied to derive an index from it. Other LEL objects
317 specifying time considerations (shipping day or maturity day, among others) could also appear in
318 the temporal location and probably become indexes.

319 **Rule 3. Scenarios' actors that have multiple instances become *indexes* of the mathematical**
320 **programming model**

321 Actors in scenarios are derived from LEL subjects. A subject in LEL could denote a specific
322 person or a role. If it is a role, which is filled by several persons, there should be an index in the
323 mathematical programming model to represent them. If the cardinality of a particular actor is likely
324 to grow from 1 into several people, the mathematical programming expert could decide to include
325 the index to make the model more flexible to accommodate this change in the near future.

326 **Rule 4. Scenarios' resources that have multiple instances become indexes of the**
327 **mathematical programming model**

328 Resources represent relevant physical elements or information used by scenarios' actors to achieve
329 their goal. Resources derive from LEL's objects. Objects can be singletons (a single instance) or
330 denote a class of elements. When an object denotes a class, it becomes an index of the
331 mathematical programming model. Similar to Rule 3, if the object could grow into a class in the
332 future, the mathematical programming expert could decide to include the index.

333 **Rule 5. Actors and/or resources which are related by a notion in the LEL become *sets* in the**
334 **mathematical programming model when their relation denotes a restriction**

335 Two or more actors and/or resources are related when they appear in the same notion. In the case
336 where the relation among them is restricted for some cases, this restriction should be defined as a
337 set. Conversely, if the relation is many-to-many, it would not be necessary to define the sets.

338 **Rule 6. The number of instances of the indexes could become parameters in the**
339 **mathematical programming model**

340 An actor that becomes an index derives from a LEL subject with multiple instances. The number
341 of instances of an index is known, and therefore, it could become a parameter of the mathematical
342 programming model, although this is not always the case. The same occurs with resources and
343 temporal data objects. An example is the number of instances of the time period, which matches
344 the decision time horizon and could be defined as a parameter.

345 **Rule 7. Attributes of scenarios' actors and attributes of scenarios' resources and attributes**
346 **of their relationship with known values become parameters of the mathematical**
347 **programming model. Each parameter is indexed by those indexes related to it by the same**
348 **notion and for which its value remains known**

349 In the case of attributes that did not become indexes by Rule 4 and denote a known value, by this
350 rule they become parameters of the mathematical programming model. Moreover, a parameter
351 derived by this rule should be indexed by the indexes that are related to it. We define two LEL
352 terms as related if they appear in the same notion, i.e., either one of them appears in the notion of
353 the other, or both terms appear in the notion of a third term. These indexes could include other
354 actors but also other resources. However, note that not all indexes that appear in the notion will be
355 assigned to the parameter, only those that refer to a known value. Thus, the mathematical
356 programming expert should determine which subset of indexes refers to a known value, and the

357 parameter should be indexed only by that subset of indexes. The notion gives the whole subset,
358 but the expert decides what indexes should be used.

359 **Rule 8. Attributes of scenarios' actors and attributes of scenarios' resources and attributes**
360 **of their relationship that have unknown values become decision variables of the**
361 **mathematical programming model. Each decision variable is indexed by those indexes**
362 **related to it by the same notion and for which its value is unknown**

363 This rule is similar to Rule 7 but for unknown values. That is, attributes that appear in the notion
364 of actors or resources, which values should be assigned in the decision process, become decision
365 variables. Moreover, decision variables should take the indexes that are related to it by the same
366 notion and refer to unknown values. It could be necessary to define artificial decision variables
367 (without economic/physical interpretation) in order to mathematically represent a reality or to
368 force some logical constraint.

369 **Rule 9. The goal of the base scenario contains the objective function**

370 The base scenario should specify in its goal attribute, the purpose of the main actor (which
371 becomes the decision maker by Rule 1) in executing the scenario. This goal is specified as a
372 complex sentence with a relative clause that starts with "so as to" followed by the verb "minimize"
373 or "maximize". From this verb to the end, this relative clause becomes the objective function.
374 Moreover, the expert may look for further details of each objective function in the notion of the
375 LEL symbols involved in the goal.

376 **Rule 10. The set of context sentences of all scenarios become the set of constraints of the**
377 **mathematical programming model**

378 Scenarios should specify in its context attribute, the conditions to comply. These conditions are
379 described in natural language, as sentences that relate to the actors and resources of each scenario.

380 Experts in mathematical programming modeling should use the restrictions described in the
 381 scenarios' context and relate them to parameters and indexes previously defined by other rules to
 382 derive the set of mathematical programming model constraints. These constraints usually contain
 383 logical constraints that represent business rules. These rules should appear during the requirements
 384 elicitation that a system analyst carries out to construct the LEL, and therefore they would also be
 385 derived as part of the scenarios' context. Additionally, there are added other artificial constraints
 386 (for instance, a positive boundary to variables) in order to avoid erroneous results. These
 387 constraints will not generally appear in the conceptual model and they should be added by a
 388 subsequent analysis of the mathematical programming expert.

389 Table 4 summarizes the relationship among LEL, scenarios and mathematical programming model
 390 elements. Rules are grouped by the mathematical programming model element that is derived from
 391 them. The second and third columns use indentation to represent nested concepts (for example, in
 392 Rule #2, the scenario element that generates an "Index" is the "Time period", inside the "Temporal
 393 location", which is in the "Context" of the "Base scenario").

394 **Table 3.** Equivalences among LEL, scenario and mathematical programming model elements.

Rule Number	LEL model	Scenario model	Mathematical model
1	Subject	Base scenario Main actor	Decision Maker
2	Object	Base scenario Context Temporal location Time period	Index
3	Subject (multiple instances or single instance--opt)	Any scenario Actor	Index
4	Object (multiple instances or single instance--opt)	Any scenario Resource	Index
5	Subject / Object Notion (restriction)	Any scenario Actor / Resource	Set
6	Subject / Object Notion (no. of instances)	Any scenario Actor / Resource (index)	Parameter
7	Subject / Object Attribute (known value)	Any scenario Actor / Resource	Parameter

Attribute			
8	Subject / Object Attribute (unknown value)	Any scenario Actor / Resource Attribute	Decision Variable
9	Subject Behavioral response (verb)	Base scenario Goal	Objective function
10	Object	Any context	Constraint

395 **4. Application**

396 This section applies the above methodology to the problem of fresh tomato packing addressed by
397 Miller et al. (1997). The purpose of using an existing problem is to contrast the result of our
398 derivation rules with a real, published mathematical programming model while avoiding the long
399 description that a new mathematical program would require.

400 **4.1 Conceptual model of the tomato packing problem: LEL + scenarios**

401 This section presents the conceptual model for the tomato packing problem created in terms of
402 LEL + scenarios. The team assembled for this task was multidisciplinary, that is, composed of
403 system analysts and a mathematical programming expert. The information sources of the UofD to
404 construct the LEL were provided by the article from Miller et al. (1997), interviews with local
405 tomato producers that played the role of customers and other documentation sources online. After
406 three iterations, the team arrived at the LEL that appears in Table 4.

407 **Table 4.** LEL of the tomato packing problem.

Term	Role	Notion
Packinghouse management (PM)	Subject	Conducts the business of the packinghouse. The PM decides when to harvest tomatoes matured in the present or previous cycle, communicates its decision to growers and packs the harvested tomatoes to fulfil the market demand
Grower	Subject	Person responsible for a tomato field (has been assigned a certain number of acres of tomatoes), including harvesting the tomatoes, for which it has some harvest capacity. There are several growers. Each grower produces a certain yield of bins of tomatoes per acre to take them to the packinghouse
Market	Subject	Customers of the packinghouse, who buy tomatoes to sell them. The market has a market demand in number of boxes of tomatoes they would like to buy each day
Acres of tomatoes	Object	Land assigned to a grower with tomatoes that get matured on a certain maturity day, and are harvested on a certain harvesting day. Acres may have a fraction of "vine ripe tomatoes" that are sold "as is"
Tomato	Object	Produce planted by growers on the acres of their fields. Tomatoes that have not been harvested in 2 cycles generate a cost of damaged tomato
Harvesting day	Object	Day in which tomatoes already matured are harvested. Every day in the horizon may be a harvesting day
Maturity day	Object	Day in a period in which tomatoes in an acre get ready to be harvested. The acres of a specific grower may have different maturity days within the decision horizon

Decision horizon	Object	Largest time in which the readiness of tomato fields for harvest can be accurately predicted. It is 3 days
Harvest capacity	Object	The capacity of a grower to harvest during a certain day
Bin	Object	Container where the grower places the harvested tomatoes
Cost of damaged tomato	Object	Penalty cost due to dissatisfaction of a grower because of delayed harvest of fields matured in the previous and present cycles in \$ per bin
Fraction of “vine ripe tomatoes” (v.r.t.)	Object	Tomatoes sold without gassing
Yield of bins of tomatoes per acre	Object	The number of bins of harvested tomatoes per acre of a certain grower
Packinghouse	Object	Place where tomatoes are packed in boxes and stored. It has a packing capacity, and a gassing capacity per day. The packinghouse requires some fraction of hour needed to pack a bin at a certain packing cost. The packinghouse generates a fraction of tomatoes ready after gassing. The packinghouse works on regular hours (9 to 5) and overtime hours (after 5pm) to generate an inventory level at the end of the day
Packing cost	Object	The total packing cost consist of: (1) cost of damaged tomatoes; (2) inventory holding cost, (3) shortfall cost, (4) overtime and (5) regular hour packing costs
Packing capacity	Object	No more tomatoes may be packed once the packinghouse reaches the packing capacity
Gassing capacity	Object	Capacity of the packinghouse to gass tomatoes in a harvesting day. It is measured in number of boxes of tomatoes
Fraction of hour needed to pack a bin	Object	Time required for packing 1 bin of harvested tomatoes and put them in boxes
Fraction of tomatoes ready after gassing	Object	Tomatoes ready after a gassing session
Regular packing hours	Object	Hours when the packinghouse is operating on a day, that generates a regular packing cost. It goes from 9 am to 5 pm
Overtime packing hours	Object	Extra hours required to complete the packing at the packinghouse on a day. They generate a higher cost than the regular packing hours. Overtime hours are after 5pm
Regular packing cost	Object	Cost of packinghouse operation in \$ per hour during regular packing hours
Overtime packing cost	Object	Cost of packinghouse operation in \$ per hour during overtime packing hours
Inventory level	Object	Number of boxes of tomatoes of the packinghouse at the end of a certain day. It has a certain inventory holding cost in box/day
Inventory holding cost	Object	Cost of packinghouse storage in boxes/day
Market demand	Object	Number of boxes of tomatoes that the packinghouse customers require in each harvesting day. If the market demand is not covered it generates a shortfall
Shortfall	Object	Number of missing boxes of tomatoes needed to reach the market demand on a certain harvesting day
Shortfall cost	Object	Cost of being short on satisfying the market demand in \$ per box short
Box	Object	Container where tomatoes are placed during packing
Harvest	Verb	To cut the tomatoes that are matured
Pack	Verb	Put in boxes the harvested tomatoes. Packing is done on regular hours at a certain cost or overtime hours to complete packing all harvested tomatoes
Gass	Verb	Technique used to ripen tomatoes that are not completely matured by exposing them to ethylene gas

408
409 The first version of the scenarios was created after the first iteration of the LEL and was used to
410 retrofit the second iteration of the LEL.
411 The following four tables describe the scenarios of the tomato packing problem. First, Table 5
412 presents the sole Level 0 scenario, called *base scenario*. Then, Tables 6 through 8 show the three
413 scenarios of Level 1, which derive from episodes of the base scenario.

414

415 **Table 5.** Scenario Level 0 of the tomato packing problem.

Scenario 0: Plan the harvest and the packing of fresh tomatoes	
Goal	Make a plan at the beginning of the present cycle to harvest and pack tomatoes matured on the present and last cycles, so as to minimize the total packing cost
Actors	Main actor Packinghouse management (PM)
	Secondary actors Growers, market
Resources	Physical resources Packinghouse; acres of matured tomatoes for each grower in the present and past cycles; tomatoes; bins; boxes.
	Information resources Market demand in number of boxes for each day in the present cycle; for each grower, the number of acres of matured tomatoes in the present and past cycles and the number of bins of tomatoes generated after harvesting.
Context	<ul style="list-style-type: none"> - Not all tomatoes that get matured in a cycle are harvested in the same cycle. - Tomatoes harvested in the next cycle after they get matured may be sold but have less quality. - Tomatoes not harvested in the next cycle after they get matured must be discarded. - A grower may only harvest up to their capacity. - The packinghouse may only pack and gass a limited number of tomato boxes a day. - Temporal location: <ul style="list-style-type: none"> • Decision horizon: 3 days • Decision period: harvesting day; every day in a decision horizon. • Other temporal variables: maturity day; any day in the decision horizon.
Episodes	<ul style="list-style-type: none"> - PM makes a plan with the harvesting day of matured tomatoes, for each day in the present cycle, in number of acres for each grower. - Each grower harvests the amount decided and communicated by the PM. - Each grower takes the harvested tomatoes in bins to the packinghouse. - PM packs the harvested tomatoes in the packinghouse, labelling some boxes as “vine-ripe tomatoes”. - PM gasses all boxes of tomatoes which are not labelled as “vine ripe”. - Tomato boxes are shipped to cover the market demand, and the surplus remains as inventory of the packinghouse.

416 **Table 6.** Scenario 1.1 at Level 1 of the tomato packing problem.

Scenario 1.1: Plan the the harvesting day of matured tomatoes	
Goal	Decide how many acres of matured tomatoes to harvest for each grower in each day of the present cycle
Actors	Main actor Packinghouse management (PM)
Resources	Physical resources Acres of tomatoes.
	Information resources For each grower: harvest capacity, acres of matured tomatoes for each day and yields of bins of tomatoes per acre. Also: cost of tomatoes damaged due to delayed harvest, packing and gassing capacity of the packing house.
Context	<ul style="list-style-type: none"> - A grower may only harvest in a day up to their harvest capacity. - The number of bins of harvested tomatoes to pack from all growers should be less than the available gassing capacity of the packinghouse for that day. - The number of bins of harvested tomatoes to be packed per day should be less than the combined regular and overtime packing capacity of the packinghouse.
Episodes	<ul style="list-style-type: none"> - Considering the information resources available, the PM calculates, for each day in the present cycle and for each grower, the number of acres to harvest of tomatoes matured of the previous and the present cycles.

417 **Table 7.** Scenario 1.2 at Level 1 of the tomato packing problem.

Scenario 1.2: Harvest	
Goal	Harvest the tomatoes and take them to the packinghouse.
Actors	Main actor Grower
Resources	Physical resources Acres of tomatoes; tomatoes; bins; packinghouse.
	Information resources Harvest capacity; acres of matured tomatoes for each day; number of acres to harvest each day
Context	<ul style="list-style-type: none"> - Tomatoes to be harvested are matured.
Episodes	<ul style="list-style-type: none"> - The grower cuts the tomatoes from the acres already matured that the PM decided to cut on the present day. - The grower places the tomatoes in bins. - The grower takes the bins to the packinghouse.

418

419 **Table 8.** Scenario 1.3 at Level 1 of the tomato packing problem.

Scenario 1.3: Pack the harvested tomatoes		
Goal	Put the tomatoes in boxes to be transported.	
Actors	Main actors	Packinghouse personnel
Resources	Physical resources	Bins of tomatoes; boxes; packinghouse.
	Information resources	Gassing capacity; regular packing hours; regular packing hour cost; overtime packing hours; overtime packing hour cost
Context	<ul style="list-style-type: none"> - Packing occurs during regular packing hour plus overtime packing hours, which combined are at most 12 hours. - Regular packing hours are at most 8 hours a day, and overtime packing hours are at most 4 hours a day. - No more tomatoes may be packed once the packinghouse reaches the packing capacity 	

420

421 **4.2 Derived mathematical programming model**

422 We show the derived mathematical programming model elements in separate tables according to
 423 the rules applied. First, Table 9 shows the decision maker and indexes generated by applying Rules
 424 1 – 4.

425 **Table 9.** Derivation of decision maker and indexes.

Rule Number	LEL model	Scenario model	Mathematical programming model
1	Subject: Packinghouse Management (PM)	Scenario 0 Main actor PM	Decision Maker <i>PM</i>
2	Object: Harvesting day	Scenario 0 Context Temporal location Decision period	Indexes <i>t</i>
2	Object: Maturity day	Scenario 0 Context Temporal location Other temporal vars	 <i>j</i>
3	Subject: Grower	Scenario 0 and 1.2 Actor: Grower	 <i>i</i>

426

427 In the process of deriving the indexes, some scenarios' resources with multiple instances were
 428 considered as candidates on which to apply Rule 4. After that, the expert reviewed the relations
 429 among actors and resources but did not derive any sets because there are none restricted relations.
 430 Applying Rule 6 to the number of instances of the index for time period *t* yielded the time horizon
 431 as the parameter *T*. The number of instances of the index for maturity day *j* is the same parameter
 432 *T*. The number of instances of the index for grower *i* yielded parameter *K*. Other parameters came
 433 from analyzing the attributes of scenarios' actors and resources. For example, the attribute acres
 434 of actor grower yielded parameter *H*. Rule 7 indicates that *H* should be indexed by the indexes

435 “related to it by the same notion”. In this case, the expert inspected the notion of acres looking the
 436 underlined terms (i.e., related elements) that were defined as indexes: i for “grower”, j for
 437 “maturity day” and t for “harvesting time”. However, the known values of acres are their grower
 438 and their maturity day, but their harvesting time is unknown. Therefore, the indexes assigned to
 439 the parameter are i and j , and the parameter is H_{ij} . Other parameters were derived similarly. The
 440 whole list of parameters appears in Table 10.

441 **Table 10.** Derivation of parameters.

Rule Number	LEL model	Scenario model	Mathematical programming model parameter
6	Object: Harvesting day Index: t	Scenario 0 Context Temporal location Decision period	T
6	Subject: Grower Index: i	Scenario 0 and 1.2 Actor: Grower	K
7	Subject: Grower Attribute: Acres Related indexes: Grower (i), Maturity day (j)	Scenario 0 and 1.2 Actor: Grower Attribute: Acres	H_{ij}
7	Subject: Grower Attr: Harvest capacity Related indexes: i	Scenario 0 and 1.2 Actor: Grower Attr: Harvest capacity	U_i
7	Subject: Grower Attr: Yields of bins Related indexes: i	Scenario 0 and 1.2 Actor: Grower Attr: Yields of bins	b_i
7	Subject: Market Attr: Market demand Related indexes: Harvesting day (t)	Scenario 0 Actor: Market Attr: Market demand	D_t
7	Object: Acres Attr: Fraction of v.r.t. Related indexes: -	Scenario 0 Resource: Acres Attr: Fraction of v.r.t.	δ
7	Object: Packinghouse Attr: Packing capacity Related indexes: -	Scenario 0 Resource: Packinghouse Attr: Packing capacity	P
7	Object: Packinghouse Attr: Gassing capacity Related indexes: t	Scenario 0 Resource: Packinghouse Attr: Gassing capacity	G_t
7	Object: Packinghouse Attr: Packing cost Notion: Cost of damaged tomato Related indexes: -	Scenario 1.3 Resource: Packinghouse Attr: Packing cost Notion: Cost of damaged tomato	C
7	Object: Packinghouse Attr: Packing cost Notion: Inventory holding cost Related indexes: -	Scenario 1.3 Resource: Packinghouse Attr: Packing cost Notion: Inventory holding cost	Ch

7	Object: Packinghouse Attr: Packing cost Notion: Shortfall cost Related indexes: -	Scenario 1.3 Resource: Packinghouse Attr: Packing cost Notion: Shortfall cost	C_s
7	Object: Packinghouse Attr: Packing cost Notion: Regular packing cost Related indexes: -	Scenario 1.3 Resource: Packinghouse Attr: Packing cost Notion: Regular packing cost	C_r
7	Object: Packinghouse Attr: Packing cost Notion: Overtime packing cost Related indexes: -	Scenario 1.3 Resource: Packinghouse Attr: Packing cost Notion: Overtime packing cost	C_o
7	Object: Packinghouse Attr: Fraction of hour needed to pack a bin Related indexes: -	Scenario 0 Resource: Packinghouse Attr: Fraction of hour needed to pack a bin	f
7	Object: Packinghouse Attr: Fraction of tom. ready after gassing Related indexes: -	Scenario 0 Resource: Packinghouse Attr: Fraction of tom. ready after gassing	α

442
443 The next step was to derive the decision variables from the analysis of the attributes of scenarios’
444 actors and resources, but this time, with unknown values. For example, the attribute acres of actor
445 grower, with a certain maturity day and with an uncertain harvesting day. This attribute yielded
446 decision variable X . Similar to the case for parameter H , to find the indexes for variable X the
447 expert looked at the underlined indexes in notion of acres, which are: i for grower, j for maturity
448 day and t for harvesting time, and the 3 of them are assigned to X to yield X_{ijt} . Other decision
449 variables were derived similarly by applying Rule 8 (I_t , R_t , O_t , S_t), and appear in Table 11. Further
450 analysis on the scenarios and the context, caused the expert to split the decision variable X_{ijt} in 2
451 variables: X_{ijt} to refer to the acres matured on the present cycle and L_{ijt} to refer to the acres matured
452 on the last cycle. Additionally, 2 more decision variables were needed to represent the acres not
453 harvested, both in the present cycle (Y_{ijt}) and the past cycle (A_{ijt}).
454 To define the objective function¹, Rule 9 was applied. Then, the expert had to manually write the
455 final equation:
456 Minimize

$$\sum_{i=1}^K \sum_{j=1}^T \sum_{t=1}^T C * (A_{ijt} + Y_{ijt}) + \sum_{t=1}^T Ch * I_t + \sum_{t=1}^T Cs * S_t + \sum_{t=1}^T Cr * R_t + \sum_{t=1}^T Co * O_t \quad (1)$$

459 **Table 11.** Resulting decision variables.

Rule	LEL model	Scenario model	Mathematical programming model
8	Subject: Grower Attribute: Acres Rel. indexes: i, j, t	Scenario 0 and 1.2 Actor: Grower Attr.: Acres	X_{ijt}
8	Object: Packinghouse Attr.: Inventory level Related indexes: t	Scenario 0 Resource: Packinghouse Attr.: Inventory level	I_t
8	Object: Packinghouse Attr.: Regular packing hours Related indexes: t	Scenario 0 Resource: Packinghouse Attr.: Regular packing hs	R_t
8	Object: Packinghouse Attr.: Overtime packing hours Related indexes: t	Scenario 0 Resource: Packinghouse Attr.: Overtime packing hs	O_t
8	Object: Market demand Attribute: Shortfall Related indexes: t	Scenario 0 Resource: Market demand Attribute: Shortfall	S_t
-	Derived manually by the expert	Scenario 0 and 1.2 Actor: Grower. Attr.: Acres (matured on <i>last</i> cycle)	L_{ijt}
-	Derived manually by the expert	Scenario 0 and 1.2 Actor: Grower. Attr.: Acres (matured <i>present</i> cycle <i>not</i> harvested)	Y_{ijt}
-	Derived manually by the expert	Scenario 0 and 1.2 Actor: Grower. Attr.: Acres (matured on <i>last</i> cycle <i>not</i> harvested)	A_{ijt}

460
461 Deriving the constraints was mostly handcrafted taking all the information available in the context
462 sentences of scenarios to create the corresponding equations, plus the addition new constraints to
463 balance quantities and make the mathematical programming model work. Constraints appear in
464 Table 12.

465 **Table 12.** Resulting constraints.

Rule	Scenario model	Mathematical programming model
10	Scenario 0 & 1.1 Context sentence: A grower may only harvest in a day up to his harvest capacity.	$\sum_{i=1}^K X_{ijt} + L_{ijt} \leq U_i \forall_{it}$
10	Scenario 1.1 Context snct: The number of bins of harvested tomatoes to pack from all growers should be less than the gassing capacity of the packinghouse for that day.	$(1 - \delta) \sum_{i=1}^K \sum_{t-j=0}^{T-1} X_{ijt} * b_i + \sum_{i=1}^K \sum_{t+j=1}^T L_{ijt} * b_i \leq G_t \forall_t$
10	Scenario 1.1 Context snct: The number of bins of harvested tomatoes to be packed per day should be less than the combined regular and overtime packing capacity of the packinghouse.	$O_t + R_t - f \sum_{i=1}^K \sum_{t-j=0}^{T-1} X_{ijt} * b_i - f \sum_{i=1}^K \sum_{t-j=1}^T L_{ijt} * b_i = 0 \forall_t$

10	Scenario 1.3 Context sntc: Packing occurs during regular packing hour plus overtime packing hours, which are at most 12 hours.	$O_t + R_t \leq 12 \forall_t$
10	Scenario 1.3 Context sntc: Regular packing hs. are at most 8 hours/day	$R_t \leq 8 \forall_t$
-	Added by expert: balance the acres matured in the previous cycle but not yet harvested	$A_{ijt} - A_{ijt-1} + L_{ijt} = 0 \forall_{ijt}$
-	Added by expert: balance the acres matured in the present cycle but not yet harvested	$Y_{ijt} - Y_{ijt-1} + X_{ijt} = H_{ij} \forall_{ijt}$
-	Added by expert: balance the end-of-period inventory level (equal to the preceding end-of-period level + the quantity of "vine ripe" tomatoes packed + the number of boxes of tomatoes ready after gassing - the forecasted demand)	$I_t - S_t - I_{t-1} + S_{t-1} - \delta \sum_{i=1}^K \sum_{t-j=0}^{T-1} X_{ijt} * b_i + \delta \sum_{i=1}^K \sum_{t-T-j=1}^T L_{ijt} * b_i = D_t + G_t \forall_t$
-	Added by expert: assure a continual flow of tomatoes to the market. The mature green tomatoes to be packed should be at least equal to a fraction α of tomatoes ready on the day of gassing.	$(1 - \delta) \sum_{i=1}^K \sum_{t-j=0}^{T-1} X_{ijt} * b_i + \sum_{i=1}^K \sum_{t+T-j=1}^T L_{ijt} * b_i \geq \alpha G_t \forall_t$

466 5. Semantic mediawiki construction for the LEL and scenarios definition

467 A domain ontology is proposed to attain the shared and agreed definition of a semantic model for
468 the LEL and scenarios. Moreover, ontology-based information retrieval allows us to formulate
469 queries based on our derivation rules, which will help in the semi-automatic derivation of
470 mathematical programming model elements. Finally, tool support is provided through a semantic
471 mediawiki constructed over the ontology, which allows for the knowledge building process of a
472 conceptual model and its derivation into math model elements.

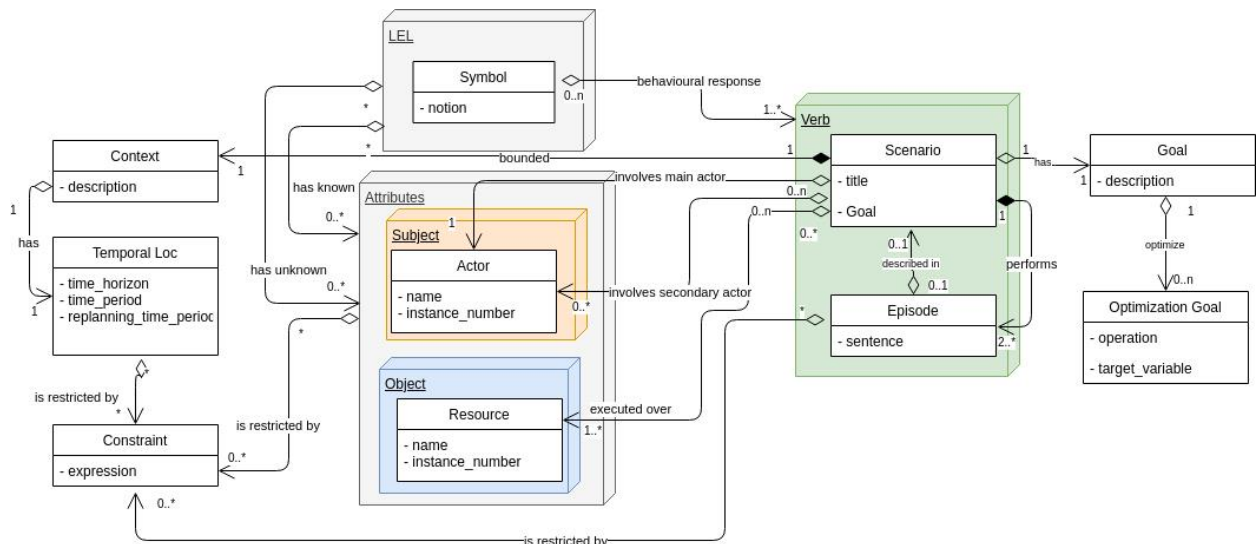
473 5.1 Ontology Model

474 The proposed ontology is depicted in Figure 2. Thus, *Subjects* symbols are related to Scenario's
475 *Actors* while *Objects* symbols are related to Scenario's *Resources*. Moreover, *Verbs* symbols are
476 related to *Scenario* and *Episode* from Scenario's representation as they represent the activities or
477 actions that are realized. LEL symbols are represented by the *Symbol* element, that has two
478 properties: a *notion* and a list of *behavioral responses*. Thus, the *notion* property just contains *has*
479 relations with *Subjects* and *Objects* that represent attributes of the described Symbol. Since we
480 need to differentiate known attributes (parameters) from unknown ones (decision variables), the
481 *has* relation is in fact modeled as two separate relations: *has known* and *has unknown*. In turn, the

482 *behavioral response* of a LEL's Symbol is represented using a relation to the *Verbs* that represent
483 the actions where the Symbol participates, and therefore, its responsibilities.

484 From the scenario's perspective, the element *Scenario* is represented by a property to define a *title*
485 as plane text. The rest of the Scenario's properties are represented as relations with other model
486 elements. Thus, there is a relation called *bounded* to a *Context*. There are two relations from
487 Scenario to the *Actor* element called *involves main actor* and *involves secondary actor*, to
488 represent the main and secondary actors of the Scenario respectively. Scenario also has a property
489 called *executed over*, to relate it with the *Resources* over which it is executed. The Scenario's
490 episodes are described by the property *performs* related with the *Episode* elements. Finally,
491 Scenarios are connected by the property *has* to the *Goal* element.

492 In turn, a *Context* has a text property *description* and a *has* property related to a *TemporalLoc* that
493 represents the temporal location of the scenarios with its properties for *time-horizon*, *time_period*
494 and *replanning_time_period*. The *Actor* element has *name* and *instance_number* as properties,
495 where the latter is used for the mathematical programming model derivation, to identify actors
496 with more than one instance as indexes. *Resource* has a structure identical to *Actor* although their
497 semantic meaning is completely different. The *Episode* element has a *sentence* that describes it
498 and a relation *described in* that connects the Episodes to the parent Scenario. Moreover, a *Goal*
499 has a property *description* and a relation *optimize* to an *Optimization Goal*. This Optimization
500 Goal is not part of the original definition of scenarios but added for mathematical programming
501 model derivation purposes. It is described by two properties: *Operation* and *TargetVariable* to
502 define the max/min operation for one particular variable as objective of the Scenario's *Goal*.
503 Finally, there is a *Constraint* element that could be related with *Episode*, *Actor*, *Resource* or
504 *TemporalLoc* elements by the property *is restricted by*. A Constraint is defined by the property
505 *expression*, which is applied on the corresponding elements.



506
507 **Fig. 2.** The LEL and scenarios ontology.

508
509 **5.2 Semantic queries**

510 The queries rely on the definition of a specific Scenario instance defined as Base Scenario. The
511 result of these queries will be used by the mathematical programming expert to obtain a
512 preliminary version of the final mathematical programming model, as explained before. Table 13
513 summarizes the queries, in a pseudo-code that makes them more readable.

514 **Table 13.** Queries based on each rule to derive potential mathematical programming model elements

Rule Number	Mathematical model output	Semantic Query
1	Decision Maker	Base Scenario involves main actor: ?Actor
2	Index	Base Scenario bounded: ?Context ?Context has: ?TemporalLoc ?time_perioc
3	Index	Scenario involves main actor: ?Actor ?Actor instance_number > 1: ?Actor Scenario involves secondary actor: ?Actor ?Actor instance_number > 1: ?Actor
4	Index	Scenario executed over: ?Resource ?Resource instance_number > 1: ?Resource
5	Set	Symbol has known: ?Attribute1 Symbol has known: ?Attribute2 (?Attribute1 is restricted by: ?Constraint) == (?Attribute2 is restricted by: ?Constraint) ?Attribute1, ?Attribute2
6	Parameter	Scenario executed over: ?Resource ?Resource instance_number > 1: ?Resource instance_number Scenario involves secondary actor: ?Actor ?Actor instance_number > 1: ?Actor instance_number

7	Parameter	(Actor has known: ?Attribute)+ (Resource has known: ?Attribute)
8	Decision Variable	(Actor has unknown: ?Attribute)+ (Resource has unknown: ?Attribute)
9	Objective function	Base Scenario has: ?Goal ?Goal optimize: ?Optimization Goal
10	Constraint	Constraint

515
516
517 Rule 1 determines the decision maker by requesting the main *Actor* from the *Base Scenario*. Rule
518 2 determines the temporal index from the time period, accessing the *Context* of the *Base Scenario*
519 through the *bounded* property, and from the *Context*, using the properties *has* to reach the
520 properties of the *Temporal Location*. Rules 3 and 4 are also intended for deriving indexes. In these
521 queries, all the instances of *Actor* and *Resource* in all the scenarios are collected by the properties
522 *involves main actor*, *involves secondary actor* and *executed over*, and they are selected if the
523 property *instance_number* is greater than 1. Rule 5 determines relations among *Actors* and
524 *Resources* that are candidates to become sets, specifically, if they appear in the same notion and
525 are related with the same *Constraint* by the property *is restricted by*. Rule 6 derives parameters
526 from the indexes obtained by rules 3 and 4, specifically, the number of instances described by the
527 property *instance_number*. Rule 7 derives parameters from known *Attributes* of *Actors* and
528 *Resources*. Particularly, this rule uses the property *has known* that belongs to *Symbol* elements
529 representing *Subjects* (for *Actors*) and *Objects* (for *Resources*). Similarly, Rule 8 determines
530 decision variables but using the property *has unknown* to denote the unknown *Attributes*. To define
531 the optimization goal of the mathematical programming model, it is necessary to access to the
532 *Goal* of the *Base Scenario* by the property *has* and subsequently, to its *Optimization Goals* by the
533 property *optimize*. Finally, any other restriction for the model could be derived from the *Constraint*
534 elements.

535 5.3 Mediawiki implementation

536 We have built a semantic mediawiki in order to provide support for the collaborative construction
537 of a knowledge base. The wiki provides the capability of creating and editing articles by way of a
538 user-friendly interface guided by forms that will be used by stakeholders, analysts and
539 mathematical modelling experts. These forms are based on the ontology proposed for the LEL and
540 scenarios. Figure 3 shows a form to describe a Scenario. The figure shows that some attributes as
541 goal and context are plain text, while others are described with a kind of button or *token*. These
542 tokens describe relations to other elements of the model already created.

Edit Scenario: Plan the harvest and the packing of fresh tomatoes

The form is titled "Edit Scenario: Plan the harvest and the packing of fresh tomatoes". It contains the following fields:

- Goal:** A text input field containing "Minimize the total packing cost".
- Context:** A text input field containing "Temporal location:
* Decision horizon: 3 days
* Decision period: harvesting day; every day in a decision horizon.
* Other temporal variables: maturity day; any day in the decision horizon."
- Main Actor:** A text input field containing "x Packinghouse management".
- Secondary Actors:** A text input field containing "x Grower".
- Resources:** A text input field containing "x Acres", "x Tomatoes", "x Bins", and "x Boxes".
- Episodes:** A text input field containing "x Each grower takes the harvested tomatoes in bins to the packinghouse."
- Exceptions:** A text input field containing "x A grower may only harvest in a day up to his harvest capacity."
"x The number of bins of harvested tomatoes to pack from all growers should be less than the gassing capacity of the packinghouse for that day."

543
544 **Fig. 3.** Mediawiki form based on LEL and scenarios' ontology.

545
546 In turn, Figure 4 shows a form to navigate a Scenario. The wiki-links could be blue or red
547 describing whether the related article is already defined or not, respectively.

Plan the harvest and the packing of fresh tomatoes

Contents [hide]

- 1 Goal
- 2 Context
- 3 Actors
- 4 Resources
- 5 Episodes
- 6 Exceptions

Goal [edit source]
Minimize the total packing cost

Context [edit source]
Temporal location: Decision horizon: 3 days...

Actors [edit source]

Main Actor	Packinghouse management
Secondary Actors	Grower

Resources [edit source]

Resources	Acres, Tomatoes, Bins, Boxes
------------------	------------------------------

548

549 **Fig. 4.** Scenario's article.

550

551 Finally, the capability of application of ontologies as a semantic knowledge model allows to
552 implement the semantic queries described previously to semi-automatically derive mathematical
553 programming model elements from LEL + scenarios elements. Thus, once the knowledge base is
554 constructed, the mathematical modelling expert will be able to generate new articles with a
555 preliminary version of the mathematical programming model. Figures 5 displays an example, with
556 the article generated from the LEL and scenarios of the case study. Note that the support is not
557 completely automatic because the inference only allows for an approximation to the mathematical
558 programming model, and the expert is still needed to verify the correctness of the mediawiki
559 derivation and provide the algebraic form.

560

Decision Maker [edit | edit source]

Packinghouse management

Objective function [edit | edit source]

Description: Minimize the total packing cost.
Type: **Minimization**
Objective variable: Packing cost

Indexes [edit | edit source]

By rule 2 [edit | edit source]

- Harvesting day
- Maturity day

By rule 3 [edit | edit source]

- Grower

By rule 4 [edit | edit source]

- Acres
- Tomatoes
- Bins
- Boxes

Sets [edit | edit source]

Constraints [edit | edit source]

From constrains expressions: [edit | edit source]

- A grower may only harvest in a day up to his harvest capacity.
- The number of bins of harvested tomatoes to pack from all growers should be less than the gassing capacity of the packinghouse for that day.
- The number of bins of harvested tomatoes to be packed per day should be less than the combined regular and overtime packing capacity of the packinghouse.
- Packing occurs during regular packing hour plus overtime packing hours, which are at most 12 hours.
- Regular packing hs. are at most 8 hours/day

Expert's considerations [edit | edit source]

Rule 4

Remove indexes:

- Acres
- Tomatoes
- Bins
- Boxes

Rule 7

Packing cost:

- Cost of damaged tomato
- Inventoring holding cost
- Shortfall cost
- Regular packing cost
- Overtime packing cost

Rule 8

Acres:

- Matured on *last* cycle

561

562

Fig. 5. Article of mediawiki with automatic mathematical programming model derivation.

563

564 **6. Conclusions**

565 This paper has presented a novel methodology that connects the areas of requirements engineering

566 with conceptual modelling in order to build mathematical programming models that capture the

567 business domain more effectively and completely. Specifically, the methodology has proposed for

568 the first time the use of the LEL and scenarios for creating a conceptual model of a domain from

569 where a mathematical programming model can be derived. The construction of the conceptual
570 model invites the participation of all stakeholders, which is deficiency of other proposals for DSS
571 construction in agriculture. In comparison with other approaches for conceptual mathematical
572 modelling, this article provides further tool-supported guidance about how to obtain the problem
573 definition and how to derive a mathematical programming model from a precise specification in
574 terms of derivation rules, as opposed to relying on mere textual descriptions or in the modeler's
575 ability. Moreover, we have proposed an ontology that provides the basis for a semantic mediawiki
576 that serves both, sharing knowledge of the conceptual domain model among the different
577 stakeholders, as well as semi-automating the derivation of the mathematical programming model.
578 The usefulness of this proposal can be understood from several perspectives: research, academic
579 and managerial. From the research and academic points of view, we may highlight the main
580 contributions as follows: (i) it provides a novel step-by-step methodology based on the LEL and
581 scenarios that allows both: to obtain the required information to derive the definition part of a
582 mathematical programming model, and to define the optimization problems that constitute the
583 modelling part of the model; (ii) our approach provides a structure to the problem that allows to
584 identify the elements of the problem clearly; (iii) using the LEL and scenarios to create a
585 conceptual model iteratively and incrementally in collaboration with stakeholders allows applying
586 an agile development approach to mathematical modelling; (iv) the use of LEL and scenarios
587 provides traceability from the requirements to the mathematical programming model
588 implementation to cope with possible changes of requirements and a better understanding of their
589 impact on the model; and (v) the process of creating a conceptual model with LEL + scenarios
590 also generates a complete specification of requirements for a potential model-based DSS.
591 Regarding the managerial perspective, we believe that the ease of use and good expressiveness of
592 the proposed methodology will facilitate the implementation of mathematical programming
593 models in agriculture, as well as provide new tools for teaching mathematical programming and

594 foster research in the combined areas of agile methods in requirements engineering, mathematical
595 programming and decision support system development. Further research includes validating the
596 proposed methodology in real world case studies from agriculture. Finally, we intend to extend
597 the approach to the derivation of mathematical programming models under uncertainty, such as
598 stochastic programming and fuzzy mathematical programming.

599 **Acknowledgments.** This work was supported by the European Commission, project H2020-
600 RUC-APS, grant number H2020-MSCA-RISE-2015-691249; and the Argentinian National
601 Agency for Scientific and Technical Promotion (ANPCyT), grant number PICT-2015-3000.

602 **References**

- 603 Alemany, M., Ortiz, A. & Fuertes-Miquel, V.S. (2018). A decision support tool for the order
604 promising process with product homogeneity requirements in hybrid Make-To-Stock and
605 Make-To-Order environments. Application to a ceramic tile company. *Computers &*
606 *Industrial Engineering* 122: 219–234.
- 607 Alemany, M.M.E., Alarcón, F., Lario, F.C. & Boj, J.J. (2011). An application to support the
608 temporal and spatial distributed decision-making process in supply chain collaborative
609 planning. *Computers in Industry* 62: 519–540.
- 610 Alemany, M.M.E., Lario, F.C., Ortiz, A. & Gómez, F. (2013). Available-To-Promise modeling
611 for multi-plant manufacturing characterized by lack of homogeneity in the product: An
612 illustration of a ceramic case. *Applied Mathematical Modelling* 37(5): 3380–3398.
- 613 Alemany, M.M.E., Pérez Perales, D., Alarcón, F. & Boza, A. (2007). Planificación Colaborativa
614 para Redes de Suministro-Distribución (RdS/D) mediante programación matemática en
615 entornos distribuidos. In *Int. Conference on Industrial Engineering & Industrial*
616 *Management*.
- 617 Alexander, I. & Maiden, N. (2004). Scenarios, stories, and use cases: the modern basis for system
618 development. *Computing Control Engineering Journal* 15(5): 24–29.
- 619 Armengol, A., Mula, J., Díaz-Madroño, M. & Pelkonen, J. (2015). Conceptual model for
620 associated costs of the internationalisation of operations. *Lecture Notes in Management and*
621 *Industrial Engineering* 181–188.
- 622 Baraniuk, R., Burrus, C., Johnson, D. & Jones, D. (2004). Sharing knowledge and building
623 communities in signal processing. *IEEE Signal Processing Magazine* 21(5): 10–16.
- 624 Behzadi, G., O’Sullivan, M.J., Olsen, T.L. & Zhang, A. (2018, September 1). Agribusiness supply
625 chain risk management: A review of quantitative decision models. *Omega (United Kingdom)*.
626 Elsevier Ltd.
- 627 Breitman, K.K. & Leite, J.C.S.P. (2003). Ontology as a requirements engineering product. In
628 *Proceedings of the IEEE Int. Conference on Requirements Engineering (RE)*.
- 629 Cid-Garcia, N.M. & Ibarra-Rojas, O.J. (2019). An integrated approach for the rectangular
630 delineation of management zones and the crop planning problems. *Computers and*
631 *Electronics in Agriculture* 164.
- 632 Cysneiros, L.M. & Leite, J.C.S.P. (2001). Using the language extended lexicon to support non-
633 functional requirements elicitation. In *Proceedings of the Workshops de Engenharia de*
634 *Requisitos, Wer’01*. Buenos Aires, Argentina.
- 635 Dominguez-Ballesteros, B., Mitra, G., Lucas, C. & Koutsoukis, N.S. (2002). Modelling and
636 solving environments for mathematical programming (MP): A status review and new

637 directions. *Journal of the Operational Research Society* 53(10 SPEC.): 1072–1092.
638 Retrieved from <https://doi.org/10.1057/palgrave.jors.2601361>

639 Estes, A., Alemany, M., Ortiz, A. & Peidro, D. (2018). A multi-objective model for inventory
640 and planned production reassignment to committed orders with homogeneity requirements.
641 *Computers & Industrial Engineering* 124: 180 – 194.

642 Estes, A., Alemany, M.M.E. & Ortiz, A. (2018). Conceptual framework for designing agri-food
643 supply chains under uncertainty by mathematical programming models. *International*
644 *Journal of Production Research* 56(13): 4418–4446.

645 Geisser, M. & Hildenbrand, T. (2006). A method for collaborative requirements elicitation and
646 decision-supported requirements analysis. In R. G. Ochoa S.F. (ed.), *Advanced software*
647 *engineering: expanding the frontiers of software technology*. Boston, MA: Springer.

648 Giannoccaro, I. & Pontrandolfo, P. (2001). Models for Supply Chain Management : A Taxonomy.
649 In *Production and Operations Management 2001.Conference POMS mastery in the new*
650 *millennium*. Orlando, FL, USA.

651 Gil, G.D., Figueroa, D.A. & Oliveros, A. (2000). Producción del LEL en un dominio técnico.
652 Informe de un caso. In *Proceedings of the Workshops de Engenharia de Requisitos, Wer'00*.
653 Rio de Janeiro, Brazil.

654 Grillo, H., Alemany, M.M.E., Ortiz, A. & Fuertes-Miquel, V.S. (2017). Mathematical modelling
655 of the order-promising process for fruit supply chains considering the perishability and
656 subtypes of products. *Applied Mathematical Modelling* 49: 255–278.

657 Grossmann, I. (2005). Enterprise-wide optimization: A new frontier in process systems
658 engineering. *American Institute of Chemical Engineers* 51(7): 1846–1857.

659 Gruber, T. (1993). A translation approach to portable ontology specifications. *Knowledge*
660 *Acquisition* 5(2): 199–220.

661 Gruber T. (1995). Toward principles for the design of ontologies used for knowledge sharing.
662 *International Journal of Human-Computer Studies* 43(43): 907–928.

663 Hernández, J.E., Mula, J., Ferriols, F.J. & Poler, R. (2008). A conceptual model for the production
664 and transport planning process: An application to the automobile sector. *Computers in*
665 *Industry* 59(8): 842–852. Retrieved from
666 <https://www.sciencedirect.com/science/article/pii/S0166361508000778>

667 Jain, R., Malangmeih, L., Raju, S.S., Srivastava, S.K., Immanulraj, K. & Kaur, A.P. (2018).
668 Optimization techniques for crop planning: A review. *Indian Journal of Agricultural*
669 *Sciences* 88(12): 1826–1835. Retrieved from
670 <https://www.researchgate.net/publication/329736318>

671 Kaplan, G., Hadad, G., Doorn, J. & Leite, J.C.S.P. (2000). Inspección del léxico extendido del
672 lenguaje. In *Proceedings of the Workshops de Engenharia de Requisitos, Wer'00*. Rio de
673 Janeiro, Brazil.

674 Laporti, V., Borges, M.R.S. & Braganholo, V. (2009). Athena: A collaborative approach to
675 requirements elicitation. *Computers in Industry* 60(6): 367–380.

676 Leite, J.C.S.P. & Franco, A.P.M. (1993). A strategy for conceptual model acquisition. In
677 *Proceedings of IEEE International Symposium on Requirements Engineering, 1993*. San
678 Diego, California: IEEE Computer Society Press.

679 Leite, J.C.S.P., Hadad, G., Doorn, J. & Kaplan, G. (2000). A Scenario Construction Process.
680 *Requirements Engineering Journal* 5(1): 38–61. Retrieved from
681 <http://link.springer.com/article/10.1007/PL00010342>

682 Lenat, D.B. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communications*
683 *of the ACM* 38(11): 33–38.

684 Lesh, R, Landau, M. & Hamilton, E. (1983). Conceptual models in applied mathematical problem
685 solving research. In R. Lesh and M. Landau (ed.), *Acquisition of Mathematics Concepts and*

- 686 *Processes*. NY: Academic Press.
- 687 Lesh, Richard. (1981). Applied mathematical problem solving. *Educational Studies in*
688 *Mathematics* 12(2): 235–264. Retrieved from <https://doi.org/10.1007/BF00305624>
- 689 Lezoche, M., Yahia, E., Aubry, A., Panetto, H. & Zdravković, M. (2012). Conceptualising and
690 structuring semantics in cooperative enterprise information systems models. *Computers in*
691 *Industry* 63(8): 775–787. Retrieved from
692 <https://www.sciencedirect.com/science/article/pii/S016636151200125X>
- 693 Liu, L., Wang, H. & Xing, S. (2019). Optimization of distribution planning for agricultural
694 products in logistics based on degree of maturity. *Computers and Electronics in Agriculture*
695 160: 1–7.
- 696 Miller, G.A. (1995). WordNet: a lexical database for English. *Communications of the ACM*
697 38(11): 39–41.
- 698 Miller, W.A., Leung, L.C., Azhar, T.M. & Sargent, S. (1997). Fuzzy production planning model
699 for fresh tomato packing. *International Journal of Production Economics* 53(3): 227–238.
700 Retrieved from <http://linkinghub.elsevier.com/retrieve/pii/S0925527397001102>
- 701 Mir, S.A., Qasim, M., Arfat, Y., Mubarak, T., Bhat, Z.A., Bhat, J.A., ... Sofi, T.A. (2015).
702 Decision support systems in a global agricultural perspective-a comprehensive review.
703 *Journal of Agriculture Sciences* 7(1): 403–415.
- 704 Moskaliuk, J., Kimmerle, J. & Cress, U. (2009). Wiki-supported learning and knowledge building:
705 effects of incongruity between knowledge and information. *Journal of Computer Assisted*
706 *Learning* 25(6): 549–561.
- 707 Mula, J., Poler, R., García-Sabater, J.P. & Lario, F.C. (2006). Models for production planning
708 under uncertainty: A review. *International Journal of Production Economics* 103(1): 271–
709 285. Retrieved from <https://www.sciencedirect.com/science/article/pii/S0925527306000041>
- 710 Mula, Josefa, Peidro, D., Díaz-Madroñero, M. & Vicens, E. (2010). Mathematical programming
711 models for supply chain production and transport planning. *European Journal of Operational*
712 *Research* 204(3): 377–390. Retrieved from <http://dx.doi.org/10.1016/j.ejor.2009.09.008>
- 713 Mundi, I., Alemany, M., Boza, A. & Poler, R. (2013). A model-driven decision support system
714 for the master planning of ceramic supply chains with non-uniformity of finished goods.
715 *Studies in Informatics and Control* 22(2): 153–16. Retrieved from
716 <https://doi.org/10.24846/v22i2y201305>
- 717 Munir, K. & Sheraz Anjum, M. (2018). The use of ontologies for effective knowledge modelling
718 and information retrieval. *Applied Computing and Informatics* 14(2): 116–126. Retrieved
719 from <https://www.sciencedirect.com/science/article/pii/S2210832717300649#b0035>
- 720 Oliveira, A. de P.A., Leite, J.C.S. do P., Cysneiros, L.M. & Cappelli, C. (2007). Eliciting Multi-
721 Agent Systems Intentionality: from Language Extended Lexicon to i* Models. In *XXVI*
722 *International Conference of the Chilean Society of Computer Science (SCCC'07)*. IEEE.
723 Retrieved from <http://ieeexplore.ieee.org/document/4396976/>
- 724 Pérez, D., Lario, F.C. & Alemany, M.M.E. (2010). Descripción detallada de las Variables de
725 Decisión en Modelos basados en Programación Matemática en un contexto de Planificación
726 Colaborativa de una Red de Suministro/Distribución (RdS/D). In *4th Int. Conf. on Industrial*
727 *Engineering and Industrial Management; XIV Congreso de Ingeniería de Organización*.
728 Donostia - San Sebastián.
- 729 Pérez Perales, D., Lario, F.C., Alemany, M.M.E. & Hernández, J. (2012). Framework for
730 Modelling the Decision. *International Journal of Decision Support System Technology* 4(2):
731 59–77. Retrieved from
732 [http://services.igi-
732 global.com/resolvedoi/resolve.aspx?doi=10.4018/jdsst.2012040104](http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jdsst.2012040104)
- 733 Pesic, M. & van der Aalst, W.M.P. (2005). Towards a reference model for work dittribution in
734 workflow management systems. In & M. N. E. Kindler (ed.), *Proceedings of the First*

735 *International Workshop on Business Process Reference Models (BPRM,05)*. Nancy:
736 University Henri Poincaré.

737 Raghunathan, S. (1996). A structured modeling based methodology to design decision support
738 systems. *Decision Support Systems* 17(4): 299–312.

739 Saranya, S. & Amudha, T. (2017). Crop planning optimization research - A detailed investigation.
740 In *2016 IEEE International Conference on Advances in Computer Applications, ICACA*
741 *2016*. Institute of Electrical and Electronics Engineers Inc.

742 Schneeweiss, C. (2003a). Distributed decision making in supply chain management. *International*
743 *Journal of Production Economics* 84(1): 71–83. Retrieved from
744 <https://www.sciencedirect.com/science/article/pii/S092552730200381X>

745 Schneeweiss, C. (2003b). Distributed decision making—a unified approach. *European Journal of*
746 *Operational Research* 150(2): 237–252. Retrieved from
747 <https://www.sciencedirect.com/science/article/abs/pii/S0377221702005015>

748 Schön, E.-M., Thomaschewski, J. & Escalona, M.J. (2017). Agile Requirements Engineering: A
749 systematic literature review. *Computer Standards & Interfaces* 49: 79–91. Retrieved from
750 <https://www.sciencedirect.com/science/article/abs/pii/S0920548916300708>

751 Shapiro, J.F. (1993). Chapter 8 Mathematical programming models and methods for production
752 planning and scheduling. *Handbooks in Operations Research and Management Science* 4:
753 371–443. Retrieved from
754 <https://www.sciencedirect.com/science/article/abs/pii/S0927050705801884>

755 Siau, K. (2004). Evaluating the Usability of A Group Support System Using Co-Discovery.
756 *Journal of Computer Information Systems* 44(2): 17–28. Retrieved from
757 <https://www.tandfonline.com/doi/abs/10.1080/08874417.2004.11647563>

758 Soto-Silva, W.E., Nadal-Roig, E., González-Araya, M.C. & Pla-Aragones, L.M. (2016, June 1).
759 Operational research models applied to the fresh fruit supply chain. *European Journal of*
760 *Operational Research*. Elsevier B.V.

761 Swartout, B., Patil, R., Knight, K. & Russ, T. (1996). Toward distributed use of large-scale
762 ontologies. In *Proc. of the Tenth Workshop on Knowledge Acquisition for Knowledge-Based*
763 *Systems*. Banff, Canada.

764 Udias, A., Pastori, M., Dondeynaz, C., Carmona Moreno, C., Ali, A., Cattaneo, L. & Cano, J.
765 (2018). A decision support tool to enhance agricultural growth in the Mékrou river basin
766 (West Africa). *Computers and Electronics in Agriculture* 154: 467–481.

767 Vicens, E., Alemany, M., Andrés, C. & Guarch, J.. (2001). A design and application methodology
768 for hierarchical production planning decision support systems in an enterprise integration
769 context. *International Journal of Production Economics* 74(1–3): 5–20. Retrieved from
770 <https://www.sciencedirect.com/science/article/pii/S0925527301001037>

771 Young, R. (2004). *The Requirements Engineering Handbook*. Artech House ISBN: 978-1-58053-
772 266-2.

773