Zhu, Jingwei; Peng, Jun; Zhang, Liang; Truong, Linh

Improving Business Process Resilience to Long-tailed Business Events via Low-code

Published in:
2022 IEEE International Conference on Web Services (ICWS)

DOI:
10.1109/ICWS55610.2022.00057

Published: 16/09/2022

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Zhu, J., Peng, J., Zhang, L., & Truong, L. (2022). Improving Business Process Resilience to Long-tailed Business Events via Low-code. In C. A. Ardagna, N. Atukorala, B. Benatallah, A. Bouguettaya, F. Casati, C. K. Chang, R. N. Chang, E. Damiani, C. G. Guegan, R. Ward, F. Xhafa, X. Xu, & J. Zhang (Eds.), *2022 IEEE International Conference on Web Services (ICWS)* (pp. 343-348). IEEE.
https://doi.org/10.1109/ICWS55610.2022.00057

Improving Business Process Resilience to Long-tailed Business Events via Low-code

Jingwei Zhu, Jun Peng, Liang Zhang
School of Computer Science, Fudan University
Shanghai Key Laboratory of Data Science
Shanghai Institute of Intelligent Electronics & Systems
Shanghai, China
{jwzhu20, jpeng21, lzhang}@fudan.edu.cn

Hong-Linh Truong
Department of Computer Science,
Aalto University
Espoo, Finland
linh.truong@aalto.fi

***Abstract*—Among different types of changes, a specific type named *long-tailed change* (LTC), induced by wide-spectrum and sporadic events (hereafter *long-tailed business events* (LBEs), poses fresh challenges to available change management solutions in business process management. The disorder in economic and social life caused by the competition of COVID-19 epidemics and countermeasures all over the world fully demonstrates the impact of this new change management problem. Based on the principle of separation of concerns, this paper proposes a systematic framework to solve the above problem. The solution consists of a low-code mechanism for process adaptation and business policy conformance. As a result, front-line practitioners can quickly react to changes by using a domain-specific language (DSL) while a corresponding verification of functional and non-functional attributes maintains compliance with business constraints. We validate the solution through a case study of an e-commerce scenario during the COVID-19 pandemic.**

***Index Terms*—Business process, long-tailed business event, low-code, domain-specific language, process adaptation.**

I. Introduction

Change management of business processes (BPs) has become an active research topic in business process management (BPM) [1]–[4] and in service-oriented computing (SOC) [5]. Although there are numerous process change management schemes, there is a lack of effective solutions to a special kind of change, i.e., *long-tailed change* (LTC), of processes caused by *long-tailed business events* (LBEs) due to their wide spectrum and sporadic nature. From the impacts of COVID-19 on the global economy, it is obvious to see the damage in functionality loss and destruction of the non-functional attributes of business processes. In a word, LTC strongly hinders business continuity and hurts BP resilience.

Consider an imaginary online grocery chaining business, named CAICAI, which provides non-staple foods for residents in many cities in China. Each CAICAI's franchisee (*front-line* in a general sense) operates independently but all franchisees share a common business process model shown as Fig. 1 (a). For quality assurance and brand control, the franchiser (*back-end* in a general sense) enforces strict governance rules on the business and handles all changes in business processes.

Everything goes smoothly until COVID-19 came. New problems have steadily emerged and almost perished the business of franchises. Because of the dynamic evolution of the epidemic as in Fig. 1 (b) and frequent adjustment of countermeasures, CAICAI's business governance scheme is paralyzed. Orders cannot be fulfilled owing to the lockdown of communities (*functional failure*). Deliveries delay from time to time due to tightened management policy on visitors (*non-functional failure*). Even worse, situations are different from city to city, e.g., in Shanghai (b) in contrast to in Beijing (c) in Fig. 1, the policies of municipal governments for epidemic control are distinct, and the pace of policy change varies. The franchiser can no longer afford to provide customized solutions for franchisees everywhere (*the LTC requirement*), and the back-end IT department is unable to modify process models in keeping up with the pace of the situation evolution (*the timeliness of solutions*).

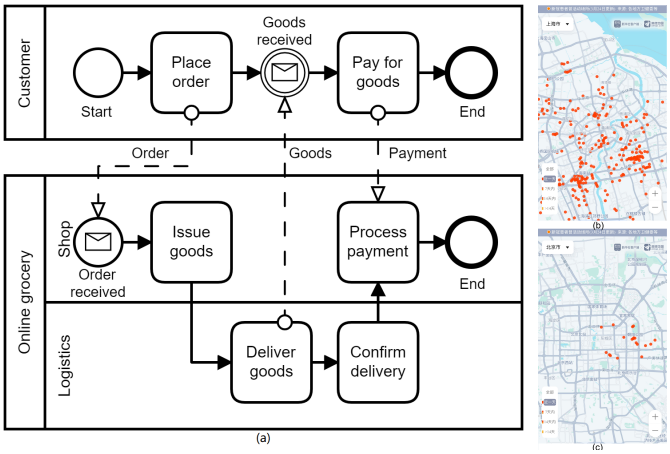

Fig. 1. The business process model of the online grocery chain store CAICAI (a), and the real-time epidemic map of Shanghai (b) and Beijing (c).

LTC is tricky because:

- Although it significantly hurts the business and induces LTC, the LBE is accidental and sporadic, which must be reacted to, but exists in too many situations. Thus, it could hardly be learned or predicted by automated analytics. COVID-19 and natural disasters are such a kind of LBE;
- LTC cannot be tackled proactively. If we could capture LBEs, corresponding LTCs are still hard to fulfill because

of their nature of wide variety in types. Too many gateways in business processes will produce spaghetti-like models that are hard to understand and govern;
- When an LTC appears, current process-aware information system (PAIS) cannot react to it timely. Classic change management techniques in BPM are too stiff to control the trade-off between flexibility and compliance.

Some of these challenges have been recognized recently, but have not been effectively addressed yet. Technology such as CMMN can adapt to new scenarios, but it heavily relies on the manual guidance of knowledge workers, which cannot meet the BPM capacity required at the production level as BPMN does. DecSerFlow [6] and its successor Declare [7] have presented extreme flexibility for process models via a declarative formal modeling language, but they are yet widely accepted by practitioners. With traditional PAIS, business practitioners, decision-makers, domain experts, and IT technicians lack novel frameworks to quickly cooperate but rely on a long bureaucratic procedure from LTC perception to business adaptation, to technical specification and deployment.

This paper focuses on issues faced by PAIS in response to LTC, and the challenge of the conflict between PAIS flexibility and compliance. We leave how to capture LBE and how to extract LTC from LBE for further research. The study contributes a set of practical techniques to improve the resilience of PAIS in the face of LTC, i.e., the front-line reacts to LTC timely via domain-specific language (DSL) annotations according to local situations, the back-end specifies core business constraints, and a tool embeds annotations into the PAIS engine and verifies constraints.

The rest of this paper is organized as follows: Section II discusses related work. Section III highlights our framework to solve resilience issues. Section IV elaborates the annotation mechanism for the front-line to react to LTCs and the constraint verification mechanism for the back-end to make reactions to LTCs comply with business regulations. Section V conducts a case study to explain the effectiveness of our framework. Section VI concludes the study.

II. Related Work

Process change/flexibility: There are plenty of studies on change/flexibility of business processes, at model level (static change) or at instance level (dynamic change), e.g., [1], [8], [9]. Two distinguished works on this topic are worth highlighting. Weber *et al.* [2] suggest a set of change patterns for modifying process models. Song and Jacobsen [5] give a comprehensive survey of process change. Our work is different from the above solutions in that we develop a low-code mechanism and a smart reaction scheme by separating the adaptation into the collaboration of front-line and back-end.

Process variability/configuration: Studies about variant families [10] or process configuration [11]–[14] can deal with process changes to some extent. They rely on a common assumption that *all* changes have to be predictable, which is not true in unpredictable LBE settings. On the contrary, our method focuses on LTC induced by LBE.

Reaction to long-tailed changes: Our early studies have noticed the existence of LBE and provided some *ad hoc* solutions, e.g., [15], [16], for LTC induced by LBE. Inspired by these previous studies, we now develop a systematic framework for LTC and a more elegant annotation implementation to react to LTC. The solution in this paper does not modify process engines, and the DSL interface is more friendly to business persons.

III. Framework Based on Separation of Concerns

Given the characteristics of LTCs, we argue that front-line business persons, rather than IT persons, are the most suitable role to react to LTCs quickly. It is of paramount importance w.r.t. costs and efficiency, if a business person can adjust the business process models once a particular LTC occurs. From the effectiveness perspective, managers of a franchise (known as process participants in [17]) are the first to perceive such changes and will eventually react to changes. In contrast, IT technicians who usually implement the business processes or administrators who govern the business cannot feel the impact of these changes unless a long period has passed. From the efficiency perspective, measures based on local conditions and contexts avoid the waste of investment in any contingency plans for LTCs. To realize this vision we develop a process behavior intervention framework[1] that allows business persons to react to LTC with low-code annotations.

A. Separation of Concerns

We apply the separation of concerns principle to resolve the conflict of flexibility and business compliance. At the front-line, business persons focus on customizing solutions to LTCs by facilities assigned; decision-makers and IT architects at the back-end are concerned about preparing these facilities and guarantee all activities are compliant with business regulations; the PAIS supports business process adaptation. As a result, the adaptation of business processes becomes the cooperation among these roles whose responsibilities are summarized in Table I. The setting greatly improves flexibility and agility.

B. Intervention in the Behavior of Business Processes

To react to LTCs, front-line business persons need to intervene in the behavior of business processes. Unlike IT persons, business persons lack the technical skills to directly modify executable process models. To enable business persons to augment the behavior of business processes, we employ an annotation mechanism. When an LTC occurs, a business person can use a DSL to annotate the process model. The annotation can intervene in the behavior of the business process by modifying the values of its process variables.

C. Compliance with Business Regulations

Compliance with business regulations, which is vital for successful execution of business processes, must still be guaranteed when dealing with LTCs. To make reactions to

[1]The source code and documents can be reached at https://github.com/SOARingLab/RESILIENCE-ICWS2022

TABLE I
RESPONSIBILITIES OF ROLES IN LTC REACTION ACCORDING TO SOC PRINCIPAL

Abbreviations: FC: functional constraint. NFC: non-functional constraints. BoD: board of directors. SA: system architect.

Role	Concern	Task	Notation
Front-line (franchise manager)	Continuity of local business	Localization of business process adaptation via annotation, API binding, localization of NFC parameters	Annotation in green and parameter in green (Fig. 4)
Back-end (franchiser, headquarters, BoD, SA, IT engineers)	Business compliance, vital business performance	DSL design, type system definition, third-party API registry establishment, FC and NFC	FC in red and NFC in orange (Fig. 4)
PAIS	Business process adaptation support	Transformation of annotation, model construction for FC and NFC, verification of FC and NFC	

LTCs comply with business regulations, we enable back-end persons to specify functional and non-functional constraints on business processes. Our framework will verify whether process models still meet these constraints after adaptation.

IV. DSL-BASED ANNOTATION AND CONSTRAINTS VERIFICATION

As a generic means, an annotation in BPMN is just a kind of auxiliary text that is not interpreted by the process engine. In [15], we use annotations on business process models to instruct a modified process engine to adjust the behavior of models in reacting to LTCs. This paper inherits the basic idea, but develops a smarter script mechanism to automatically explain the annotations added by front-line business persons in DSL snippets, which can intervene in business process behavior but with a more elegant non-intrusive reaction to LTC.

Compared to directly modifying the process model, adding DSL annotations has two advantages. First, DSL is more user-friendly for business persons, so the front-line can deal with LTCs more agilely. Second, it is lightweight to match the nature of LTCs that adapts to current situations and is easy to drop changes afterward. It is flexible enough while keeping the stability of process models to a certain extent.

A. Grammar of DSL

Back-end IT persons design a DSL for their business. For example, the back-end of CAICAI designs a DSL, whose grammar is shown in Fig. 2. The DSL can specify, e.g., when a certain condition is met, some actions should be done to respond to the condition.

The current status of a process instance is reflected by the values of its process variables. In the DSL, a condition is expressed in whether process variables have certain values. The values of process variables determine the execution behavior of the process instance in the future. In the DSL, an action is an assignment of a value to a variable.

B. Value Ranges of Process Variables

Front-line business persons intervene in the behavior of a business process by modifying the values of its process variables. In order for the process to execute correctly, process variables have certain value ranges. Back-end persons specify the value ranges of process variables, and front-line persons can specify more concrete value ranges according to their local situations. For example, the value ranges of process variables in CAICAI's process are shown in Table II.

```
code = { sentence } ;
sentence = "WHEN" , condition ,
  { "SET" , action } ;
condition = variable , ( "==" | "!="
  | "<" | "<=" | ">" | ">=" ) , value
  | condition , "AND" , condition
  | condition , "OR" , condition ;
action = variable , "=" , value ;
variable = ? variable identifier ? ;
value = ? literal value ? | variable
  | value , ( "*" | "/" ) , value
  | value , ( "+" | "-" ) , value
  | "(" , value , ")" ;
```

Fig. 2. Grammar of the DSL for CAICAI.

TABLE II
THE VALUE RANGES OF PROCESS VARIABLES IN CAICAI'S PROCESS

Process variable	Type	Value range
order_status	String	"pending", "finished", or "canceled"
region	String	Non-empty strings
delivery_method	String	"home_delivery", "contactless_locker", or "at_store"
amount	Number	Decimals greater than 0

C. Binding Third-party APIs

Some third-party APIs provide information about the current situation and local contexts that is helpful in responding to LTCs. A blog [18] introduces a method of calling public APIs without programming. We employ this method to let front-line business persons bind third-party APIs. A front-line person can bind a third-party API by specifying the URL of the API, variables as arguments, and variables to store return values. In addition, back-end persons can establish an API repository to guide front-line persons to third-party APIs.

D. Enhanced Process Modeler

In order to facilitate front-line persons to annotate process models with DSL, we develop an enhanced process modeler based on bpmn.io. Inspired by [16], we design a syntax-directed editor for writing DSL codes, so that the syntax correctness of DSL codes can be guaranteed.

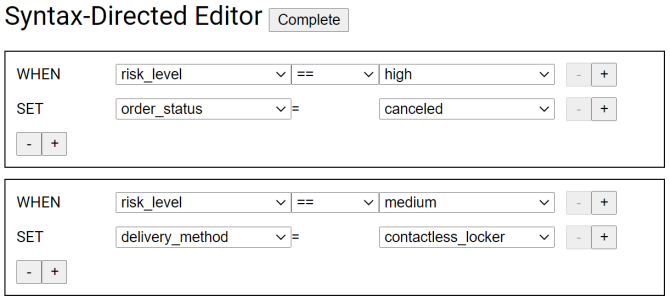

Fig. 3. Interface for specifying change annotations based on DSL.

A front-line person can choose an activity and add a text annotation to it. Then, the business person can write a DSL code on the text annotation using the syntax-directed editor, as shown in Fig. 3.

For example, a franchise manager of CAICAI decides to annotate the process model, so that the delivery of goods can be adjusted according to the risk level of the region the customer is in:

- If the risk level is high, goods cannot be delivered at all, and the order has to be canceled.
- If the risk level is medium, goods will be placed in a contactless locker near the customer.
- If the risk level is low, goods will be delivered to the customer's home as usual.

The franchise manager adds a text annotation to the activity *Issue goods*, in order to adjust the delivery of goods, as shown in Fig. 4. The risk level is obtained from a third-party API and is stored in the variable *risk_level*. If its value is "high", the variable *order_status* should be set to "canceled", so that the order will be canceled; if its value is "medium", the variable *delivery_method* should be set to "contactless_locker", so that goods will be placed in a contactless locker.

E. Process Model Transformer

Typically, process engines will ignore text annotations while executing a process model. We develop a process model transformer to let annotations go into effect. The transformer can transform a process model annotated with DSL into an ordinary process model, so that existing process engines can perceive the semantics of text annotations when executing the process model. During transformation, each text annotation will be converted into a script task placed before the activity to which the text annotation is added.

The process model transformer includes a DSL code translator which can translate a DSL code into a Groovy code. The Groovy code will be embedded into a script task. Groovy codes embedded in script tasks can be executed by a process engine, such as Camunda or Activiti.

The process model transformer works independently of process engines. With the help of the transformer, we can continue to use existing process engines directly, instead of modifying existing process engines or developing new process engines. In contrast to the method in [15], which has to modify

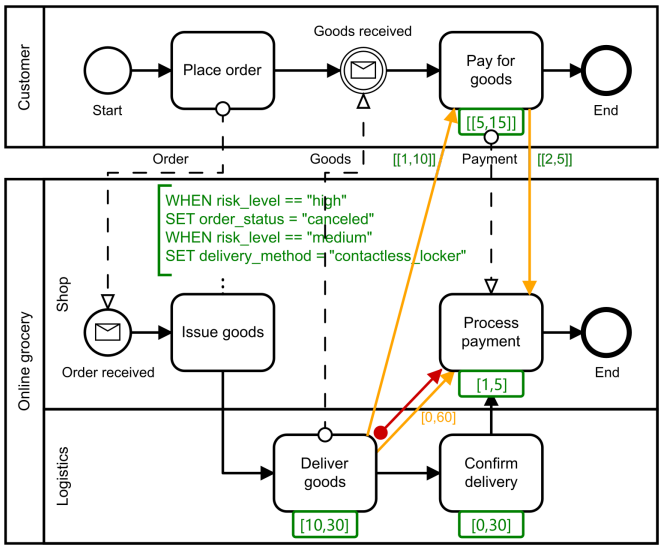

Fig. 4. The process model of CAICAI with annotation, a set of functional constraints, and a set of non-functional constraints. Green parts are customized by the front-line; red and orange ones are specified by the back-end.

process engines, our framework has a significant advantage of non-invasive enhancement of BPM systems.

F. Constraints and Verification

The compliance is enforced by modeling and verifying functional and non-functional constraints attached to annotated business process models.

Functional constraints: We employ DecSerFlow [6] to model functional constraints, i.e., the execution relations between activities. For example, the back-end specifies the functional constraint *payment must be fulfilled once goods are delivered* with the notation Deliver goods $\rightarrow$ Process payment that is equivalent to the LTL formula $\Box(\text{Deliver\_goods} \rightarrow \Diamond \text{Process\_payment})$, as depicted as the red arrow in Fig. 4. Then we can verify if all executable paths fulfill the constraint.

Non-functional constraints: Non-functional constraints and verification are delegated to G-STNU [19], e.g., the green brackets customized by a franchisee and orange texts and arrows specified by the franchiser in Fig. 4. Controllable temporal constraints are expressed with single brackets such as $[10, 30]$, whereas uncontrollable ones are expressed with double brackets such as $[[1, 10]]$. Then we can test if we could set the controllable ones to meet all constraints regardless of those uncontrollable ones. Note that besides time constraints by STNU [20], G-STNU supports more general qualitative and quantitative constraints. Details are refereed to [19].

V. Case Study

We conduct two case studies for the proposed solution, one to evaluate the effectiveness/efficiency, and the other to validate the applicability for multiple open-source BPM suites.

Effectiveness/efficiency validation: We emulate different situations for CAICAI to fight against the situation of the Omicron outbreak in Shanghai since March 2022.

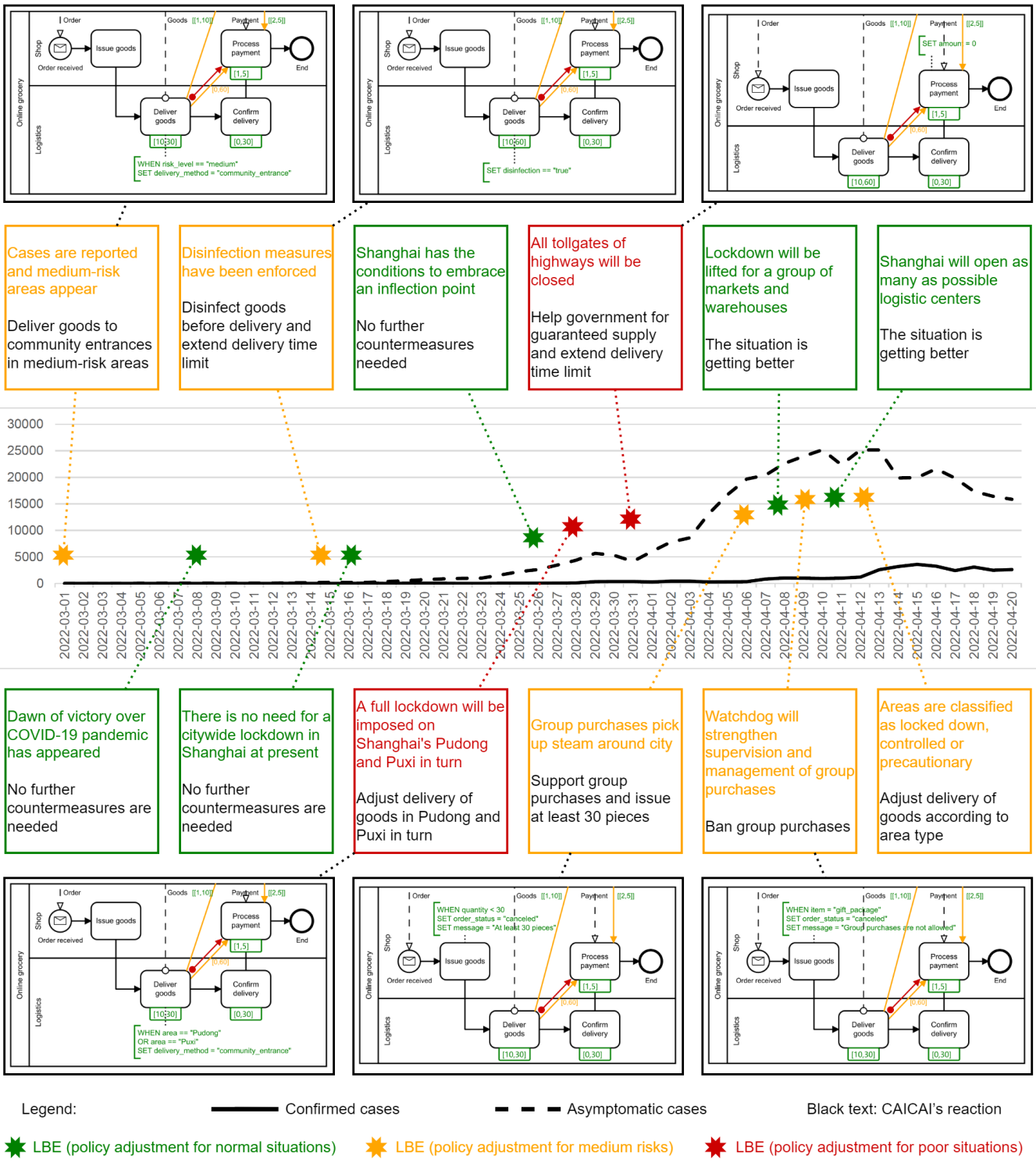

Fig. 5. Illustration of epidemic evolution and corresponding adjustments of countermeasure policies in Shanghai, and CAICAI's adaptation actions for its business process:
(1) The timeline in the middle shows the development in the number of asymptomatic infections (black dash line) and confirmed infections (black solid line) of Omicron officially reported in Shanghai from 2022-03-01 to 2022-04-20. The Omicron outbreak resulted in the long-tailed business event (LBE), the frequent adjustment of countermeasure policies (texts in green, orange, or red in boxes) poses a series of long-tailed change (LTC) requirements (marked as explosion marks at the middle area) that disrupt CAICAI's original franchising business due to various failures, e.g., orders cannot be fulfilled owing to the lockdown of communities (*the functional failure*), deliveries delay from time to time due to tightened management policy on visitors (*the non-functional failure*), the franchiser is unable to modify process models in keeping up with the pace of the situation evolution (*the timeliness failure*).
(2) CAICAI's customized adaptation agilely by each franchisee according to the local situations leads to various change decisions depicted in text boxes linked to events.
(3) CAICAI changes its business processes (in the top and bottom parts of the figure) and controls its business compliance via the functional constraints (in red arrows between tasks) and non-functional ones (in yellow arrows among tasks) specified by the franchiser. The DSL annotation enables each franchisee to react in time to LTC and carries out her business smoothly. In general, the proposed method enhances the resilience of CAICAI's PAIS.
Further details of the example can be referred to the GitHub repository of our project.

With the development of the epidemic situation (see Fig. 5), the epidemic counter policy of the municipal government is also changing rapidly, from the initial non-interference with daily life to the temporary closure of certain buildings or communities, to the separated treatment under the lockdown of the whole city on April 1 (see Fig. 5). On the contrary, Wuhan, which had the greatest impact of COVID-19 in 2020, did not have any infection. Other cities (such as Beijing in Fig. 1 (c)) had only sporadic infection cases, and the prevention and control measures were much looser.

CAICAI decided to change its business governance to the cooperation of the franchiser and franchisees: according to the solution proposed in this paper, each local franchisee can tailor her business processes through DSL annotation in response to her perception of LBEs, including the adjustment of process execution path and non-functional attributes (e.g., extending delivery time). The franchiser focuses only on regulating the functional and non-functional constraints on CAICAI's core business and verifying the compliance of the process customized by franchisees. Some LBEs and corresponding reactions are shown in Fig. 5. As a result, CAICAI obtains the following benefits from the proposed solution:

- Franchise managers work on the front line. They can sense the change in epidemic counter policies in the first place. As an ancient Chinese saying goes: the duck knows first when the river becomes warm in spring.
- Since franchise managers actually live in their cities, they can get first-hand experience of the exact epidemic counter policies. They know isolation policies, which communities are closed, etc.
- Each franchise manager is only responsible for reacting to LTCs in his or her own franchise. The effort to react to LTCs is distributed to each franchise manager. He or she has enough time to frequently adapt the process model.
- Adapting via annotation is lightweight. It makes process models more stable. Modification of process variables is limited to their value ranges. Functional and non-functional constraints are imposed on process models. Correctness of adaptation can be easily ensured.

Applicability validation: We have transported the business processes of CAICAI from Camunda to other popular open-source business process engines, i.e., Activiti and jBPM. Despite some syntactic differences, our solution can be successfully applied to all the three platforms. Whenever an LTC occurs, annotations attached to deployed process models can react to LTCs effectively and agilely, which demonstrates the excellence of the solution.

VI. Conclusion and Future Work

This paper develops an adaptation framework for PAIS to LTC induced by LBE. When an LTC occurs, front-line business persons can use a DSL to annotate process models and change the process behavior via modifying process variables in addressing LBE. Based on the SoC principle, the framework also helps back-end experts to impose functional and non-functional constraints on business processes. In a word, this approach neatly balances the conflict between flexibility and compliance of the process models.

In the future, we will enhance the approach in two directions, i.e., leveraging the framework to support process instance adaptations and supporting the negotiation mechanism in constraint specification and verification.

Acknowledgement: This work is supported by Projects of International Cooperation and Exchanges NSFC-DFG (Grant No. 62061136006)

References

[1] W. M. Van Der Aalst and T. Basten, "Inheritance of workflows: an approach to tackling problems related to change," *Theoretical computer science*, vol. 270, no. 1-2, pp. 125–203, 2002.

[2] B. Weber, M. Reichert, and S. Rinderle-Ma, "Change patterns and change support features–enhancing flexibility in process-aware information systems," *Data & knowledge engineering*, vol. 66, no. 3, pp. 438–466, 2008.

[3] W. Xu, J. Su, Z. Yan, J. Yang, and L. Zhang, "An artifact-centric approach to dynamic modification of workflow execution," in *Procs. of CoopIS'11*. Springer, 2011, pp. 256–273.

[4] M. Reichert and B. Weber, *Enabling Flexibility in Process-Aware Information Systems*. Springer, 2012.

[5] W. Song and H. Jacobsen, "Static and dynamic process change," *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 215–231, 2018.

[6] W. M. van der Aalst and M. Pesic, "Decserflow: Towards a truly declarative service flow language," in *Procs. of WS-FM'06*. Springer, 2006, pp. 1–23.

[7] W. M. van der Aalst, M. Pesic, and H. Schonenberg, "Declarative workflows: Balancing between flexibility and support," *Computer Science - Research and Development*, vol. 23, no. 2, pp. 99–113, 2009.

[8] F. Casati, S. Ceri, B. Pernici, and G. Pozzi, "Workflow evolution," *Data & Knowledge Engineering*, vol. 24, no. 3, pp. 211–238, 1998.

[9] M. Reichert, S. Rinderle, U. Kreher, and P. Dadam, "Adaptive process management with adept2," in *Procs. of ICDE'05*. IEEE, 2005, pp. 1113–1114.

[10] A. Schnieders and F. Puhlmann, "Variability mechanisms in e-business process families," in *Procs. of BIS'06*. DBLP, 2006, pp. 583–601.

[11] M. La Rosa, M. Dumas, A. H. Ter Hofstede, and J. Mendling, "Configurable multi-perspective business process models," *Information Systems*, vol. 36, no. 2, pp. 313–340, 2011.

[12] A. Hallerbach, T. Bauer, and M. Reichert, "Capturing variability in business process models: the provop approach," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 22, no. 6-7, pp. 519–546, 2010.

[13] M. Moon, M. Hong, and K. Yeom, "Two-level variability analysis for business process with reusability and extensibility," in *Procs. COMPSAC'08*. IEEE, 2008, pp. 263–270.

[14] M. L. Rosa, W. M. V. D. Aalst, M. Dumas, and F. P. Milani, "Business process variability modeling: A survey," *ACM Computing Surveys*, vol. 50, no. 1, pp. 1–45, 2017.

[15] X. Chen, H. Cao, L. Ye, and L. Zhang, "Fulfilling functional demands of bpm in long-tailed change environments," in *Procs. of CCF ICSS'20*, 2020.

[16] H. Cao, X. Chen, L. Zhang, T. Zhang, and X. Xiao, "Guaranteeing sound reactions to long-tailed changes a syntax-directed annotation approach," in *Procs. of CCF ICSS'20*, 2020.

[17] M. Weske, *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag Berlin Heidelberg, 01 2019.

[18] B. Silver, "Call public apis without programming," Jan 2022. [Online]. Available: https://www.trisotech.com/call-public-apis-without-programming/

[19] J. Peng, J. Zhu, and L. Zhang, "Generalizing stnu to model non-functional constraints for business processes," in *Procs. of CCF ICSS'22*, 2022.

[20] J. Eder, M. Franceschetti, and J. Köpke, "Controllability of business processes with temporal variables," in *Procs. of SAC'19*. ACM, 2019, pp. 40–47.