Time and Exceptional Behavior in Multiparty Structured Interactions

Hugo A. López[1] and Jorge A. Pérez[2]

[1] PLS, IT University of Copenhagen
[2] CITI - Departamento de Informática, FCT - New University of Lisbon

Abstract. The Conversation Calculus (CC) is a model of multiparty interactions which extends the π-calculus with the notion of *conversation*—a possibly distributed medium in which participants may communicate. Here we study the interplay of time and exceptional behavior for models of structured communications based on conversations. We propose C3, a *timed* variant of the CC in which conversations feature both standard and exceptional behavior. The exceptional behavior may be triggered either by the passing of time (a timeout) or by an explicit signal for conversation abortion. By presenting a compelling example from a healthcare scenario, we argue that the combination of time and exceptional behavior leads to more meaningful models of structured communications.

1 Introduction

This paper is an initial step in understanding how *time* and forms of *exceptional behavior* can be jointly captured in models of multiparty structured communications.

Time usually plays a crucial rôle in practical scenarios of structured communication. Consider, for instance, a web banking application: interaction between a user and her bank generally takes place by means of a secure session, which is meant to expire after a certain period of inactivity of the user. When that occurs, she must exhibit again her authentication credentials, so as to initiate another session or to continue with the expired session. In some cases, the session (or parts of it) has a *predetermined duration* and so interactions may also be bounded in time. The user may need to reinitiate the session if, for instance, her network connection is too slow. Crucially, the different incarnations of time in interactions (session durations, timeouts, delays) can be seen to be closely related to the behavior of the system in exceptional circumstances. A specification of the web banking application above would appear incomplete unless one specifies how the system should behave when, e.g., the session has expired and the user attempts to reinitiate it, or when interaction is taking longer than anticipated.

In real scenarios of structured communications, time then appears to go hand in hand with *exceptional behavior*. This observation is particularly evident in *healthcare scenarios* [20]—a central source of motivation for our work. In healthcare scenarios, structured communications often involve *strict time bounds*, as in, e.g., "monitor the patient every two hours, for the next 48 hours". They may also include interaction patterns defined as both a *default behavior* and an *alternative behavior* to be executed in case of unexpected conditions, as in, e.g., "contact a substitute doctor if the designated

$$
\begin{aligned}
&\mathsf{Buyer} \blacktriangleleft [\mathbf{new}\ \mathsf{Seller} \cdot \mathtt{BuyService} \Leftarrow \mathtt{buy}^{\downarrow}!(prod).\mathtt{price}^{\downarrow}?(p).\mathtt{details}^{\downarrow}?(d)] \\
&\mid \mathsf{Seller} \blacktriangleleft [\\
&\qquad \mathsf{PriceDB} \\
&\qquad\ \mid\ \mathbf{def}\ \mathtt{BuyService} \Rightarrow \mathtt{buy}^{\downarrow}?(prod).\mathtt{askPrice}^{\uparrow}!(prod).\mathtt{priceVal}^{\uparrow}?(p).\mathtt{price}^{\downarrow}!(p). \\
&\qquad\qquad\qquad\qquad \mathbf{join}\ \mathsf{Shipper} \cdot \mathtt{DelivService} \Leftarrow \mathtt{product}^{\uparrow}!(prod)] \\
&\mid \mathsf{Shipper} \blacktriangleleft [\mathbf{def}\ \mathtt{DelivService} \Rightarrow \mathtt{product}^{\downarrow}?(p).\mathtt{details}^{\downarrow}!(data)]
\end{aligned}
$$

Fig. 1: The purchasing scenario in CC.

doctor cannot be reached within 15 minutes". Also, scenarios may involve tasks that may be *suspended* or *aborted*, as in, e.g.,"stop administering the medicine if the patient reacts badly to it".

Unfortunately, expressing appropriately the interplay of time and exceptional behavior in known formalisms for structured communications turns out to be hard. In fact, although some of such formalisms have been extended with constructs for exceptional behavior (see, e.g., [10,4,7]), to the best of our knowledge none of these works considers the crucial interplay with timed behavior. To overcome this lack, here we introduce C3, a model of structured communications that integrates time and exceptional behavior in the context of multiparty interactions. C3 arises as an extension of the Conversation Calculus (CC) [24,23], a variant of the π-calculus [21] enhanced with the notion of *conversation*—a possibly distributed medium in which participants may communicate. In C3, conversations have durations and are sensible to compensations. Below, we first present the CC by means of a running example; then, we introduce C3 by enhancing the example with time and exceptional behavior.

The CC is an interesting base language for our study. First, being an extension of the π-calculus, the CC is a fairly *simple* and *elegant* model. As such, the definition of C3 can take advantage of previous works on extensions of the π-calculus with time and forms of exceptional behavior (see, e.g., [1,11]). Second, the CC counts with a number of *reasoning techniques* to build upon, in particular so-called *conversation types* [5]. Third, and most importantly, the CC allows for the specification of *multiparty interactions*, which are ubiquitous in many practical settings.

Fig. 1 gives a CC specification of the well-known *purchasing scenario* [9,23], which describes the interaction of a buyer and a seller for buying a given product; the seller later involves a shipper who is in charge of delivering the product. In the CC, a *conversation context* represents a distributed communication medium where two or more partners may interact. Process $n \blacktriangleleft [P]$ is the conversation context with behavior P and identity n; process P may seamlessly interact with processes contained in other conversation contexts named n. The model in Fig. 1 thus involves three participants: Buyer, Seller, and Shipper. Buyer invokes a new instance of the `BuyService` service, defined by Seller. As a result, a *conversation* on a fresh name is established between them; this name can then be used to exchange information on the product and its price (the latter is retrieved by Seller from the database PriceDB). When the transaction has been agreed, Shipper joins in the conversation, and receives product information from Seller and delivery details from Buyer. The model in Fig. 1 relies on the following *service idioms* which, interestingly, can be derived from the basic syntax of the CC:

$\mathbf{def}\, s \Rightarrow P$	$\triangleq$	$\mathtt{s}^{\downarrow}?(x).x \blacktriangleleft [P]$	define a service s with behavior P
$\mathbf{new}\, n \cdot s \Leftarrow Q$	$\triangleq$	$(\nu c)(n \blacktriangleleft [\mathtt{s}^{\downarrow}!(c)] \mid c \blacktriangleleft [Q])$	create instance of a service s located at n
$\mathbf{join}\, n \cdot s \Leftarrow Q$	$\triangleq$	$\mathbf{this}(x).(n \blacktriangleleft [s^{\downarrow}!(x)] \mid Q)$	join instance of service s located at n

Above, $\mathtt{s}^{\downarrow}?(x)$ and $\mathtt{s}^{\downarrow}!(c)$ are *directed* input and output prefixes (as in the π-calculus), and the $\mathbf{this}(x)$ prefix binds the enclosing conversation context to name x. The *message direction* $\downarrow$ (resp. $\uparrow$) decrees that the action should take place in the *current* (resp. *enclosing*) conversation context. We use $\star\,\mathbf{def}\, s \Rightarrow P$ to the denote an idiom for *persistent* service definition, which can be defined using recursion.

The main design decision in defining C3 is considering time and exceptional behavior *directly* into conversations: C3 features *timed*, *compensable* conversation contexts, denoted as $n \blacktriangleleft [P\,;\, Q]^{t}_{\kappa}$. As before, n is the identity of the conversation context. Process P describes the *default* behavior for n, which is performed while the *duration* t is greater than 0. Observable actions from P witness the time passage in n; as soon as $t = 0$, the default behavior is dynamically replaced by Q, the *compensating* behavior. Name κ represents an explicit *abort* mechanism: the interaction of $n \blacktriangleleft [P\,;\, Q]^{t}_{\kappa}$ with a *kill process* on κ, written $\kappa^{\dagger}$, immediately sets t to 0.

An immediate and pleasant consequence of our extended conversation contexts is that the signature of the service idioms given above can be extended too. Hence, C3 specifications can express rich information on timeouts and exceptional behavior. It suffices to extend the idioms representing *timed* service definition and instantiation:

$\mathbf{def}\, s\, \mathbf{with}\, (\kappa, t) \Rightarrow \{P; Q\}$	$\triangleq$	$\mathtt{s}^{\downarrow}?(y).y \blacktriangleleft [P\,;\, Q]^{t}_{\kappa}$	service definition
$\mathbf{new}\, n \cdot s\, \mathbf{with}\, (\kappa, t) \Leftarrow \{P; Q\}$	$\triangleq$	$(\nu c)(n \blacktriangleleft [\mathtt{s}^{\downarrow}!(c)] \mid c \blacktriangleleft [P\,;\, Q]^{t}_{\kappa})$	service instantiation

In the former we assume y, and c are fresh in P and Q, and different from κ, t, n, while in the latter $n \blacktriangleleft [\mathtt{s}^{\downarrow}!(c)]$ stands for $n \blacktriangleleft [\mathtt{s}^{\downarrow}!(c)\,;\, \mathbf{0}]^{\infty}_{\emptyset}$. This way, we are able to define timed, compensable extensions for service definition and instantiation idioms; they rely on a compensation signal κ, a timeout t, and a compensating protocol definition Q. As a simple example, the C3 processes

$$\mathsf{Provider} \blacktriangleleft [\mathbf{def}\ \mathtt{MyService}\ \mathbf{with}\ (\kappa_p, t_p) \Rightarrow \{R; T\}]$$
$$\mathsf{Client} \blacktriangleleft [\mathbf{new}\ \mathsf{Provider} \cdot \mathtt{MyService}\ \mathbf{with}\ (\kappa_c, t_c) \Leftarrow \{P; Q\}]$$

may interact and evolve into $(\nu s)(\mathsf{Provider} \blacktriangleleft [s \blacktriangleleft [R\,;\, T]^{t_p}_{\kappa_p}] \mid \mathsf{Client} \blacktriangleleft [s \blacktriangleleft [P\,;\, Q]^{t_c}_{\kappa_c}])$.

Some related approaches (e.g. [10]) distinguish the behavior originated in the standard definition of a service from the behavior associated to related compensating activities. In those works, the objective is to return to the standard control flow by orderly escaping from compensating activities; handling nested compensations thus becomes a delicate issue. In contrast, we do not enforce such a distinction: we believe that in many realistic scenarios the main goal is timely availability of services; hence, the actual origin of the offered services should be transparent to the users. This way, e.g., for the users of a web banking application, interacting with the main server or with one of its mirrors is unimportant as long as users are provided with the required services.

We illustrate these ideas by considering an extended purchase scenario in C3; see Fig. 2. Suppose a buyer who is willing to interact with a specific provider only for a

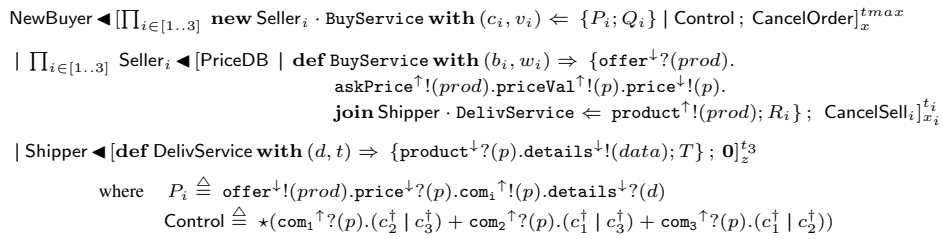

Fig. 2: The purchasing scenario in C3.

finite amount of time. She first engages in conversations with several providers at the same time; then, she picks the provider with the best offer, abandoning the conversations with the other providers. In the model, $\star\, P$ denotes the replicated version of process P, with the expected semantics. We consider one buyer and three sellers. NewBuyer creates three instances of the `BuyService` service, one from each seller. The part of each such instances residing at NewBuyer can be aborted by suitable messages on c_i. The part of the protocol for `BuyService` that resides at NewBuyer is similar as before, and is extended with an output signal $\mathtt{com}_i$ which allows to commit the selection of seller i. The commitment to one particular seller (and the discard of the rest) is implemented by process Control. The duration of NewBuyer is given by $tmax$; its compensation activity (CancelOrder) is left unspecified. Seller_i follows the lines of the basic scenario, extended with compensation signals y_i which trigger the compensation process $\mathsf{CancelSell}_i$. Notice that while Q_i controls the failure of the i-th service invoked by NewBuyer, CancelOrder is meant to control the failure of NewBuyer as a whole.

This extended example illustrates two of the features of C3: explicit conversation abortion and conversations bounded in time. The first one can be appreciated in the selection implemented by Control, which ensures that only one provider will be able to interact with NewBuyer, by explicitly aborting the conversations at NewBuyer with the other two providers. However, Control only takes care of the interactions at the buyer side; there are also conversation pieces at each Seller_i, which are not under the influence of Control (we assume $c_i \neq w_i$). The "garbage-collection" of such pieces is captured by the second feature: since such conversations are explicitly defined with the time bound w_i, they will be automatically collected (i.e. aborted) after w_i time units. That is, the passing of time avoids "dangling" conversation pieces. This example reveals the complementarity between the explicit conversation abortion (achieved via abortion signals) and the more implicit conversation abortion associated to the passing of time.

The rest of the paper is organized as follows. In Section 2 we summarize the main definitions of the CC. Section 3 introduces the syntax and semantics of C3. Section 4 discusses the expressiveness of C3, by comparing it to some other related languages. Then, in Section 5, we present a compelling example of C3 in a healthcare scenario. We review related work in Section 6; some concluding remarks are given in Section 7.

2 The Conversation Calculus (CC)

Here we briefly introduce the Conversation Calculus (CC, in the following). The interested reader is referred to [24,23] for further details.

The CC corresponds to a π-calculus with *labeled* communication and extended with *conversation contexts*. A conversation context can be seen as a medium in which interactions take place. It is similar to sessions in service-oriented calculi (see [12]) in the sense that every conversation context has an unique identifier (e.g.: an URI). Interactions in CC may be intuitively seen as communications in a pool of messages, where the pool is divided in areas identified by conversation contexts. Multiple participants can access many conversation contexts concurrently, provided they can get hold of the name identifying the context. Moreover, conversations can be nested multiple times (as in, e.g., a private chat room within a multi-user chat application).

Definition 1 (CC **Syntax).** *Let $\mathcal{N}$ be an infinite set of names. Also, let $\mathcal{L}$, $\mathcal{V}$, and χ be infinite sets of labels, variables, and recursion variables, respectively. Using d to range over $\uparrow$ and $\downarrow$, the set of actions α and processes P is given below:*

$$\alpha ::= \mathtt{l}^d!(\vec{n}) \mid \mathtt{l}^d?(\vec{x}) \mid \mathbf{this}(x) \quad P, Q ::= n \blacktriangleleft [P] \mid \sum_{i \in I} \alpha_i.P_i \mid P \mid Q \mid (\nu n)P \mid \mu X.P \mid X$$

Above, $\vec{n}$ and $\vec{x}$ denote tuples of names and variables in $\mathcal{N}$ and $\mathcal{V}$, respectively. Actions can be an output $\mathtt{l}^d!(\vec{n})$ or an input $\mathtt{l}^d?(\vec{x})$, as in the π-calculus, with $\mathtt{l} \in \mathcal{L}$ in both cases. The *message direction* $\downarrow$ (read "here") decrees that the action it is associated to should take place in the *current* conversation context, while $\uparrow$ (read "up") decrees that the action should take place in the *enclosing* one. We often omit the "here" direction, and write $\mathtt{l}?(y).P$ and $\mathtt{l}!(\vec{n}).P$ rather than $\mathtt{l}^{\downarrow}?(y).P$ and $\mathtt{l}^{\downarrow}!(\vec{n}).P$. The context-aware prefix $\mathbf{this}(x)$ binds the name of the enclosing conversation context to x. The syntax of processes includes the conversation context $n \blacktriangleleft [P]$, where $n \in \mathcal{N}$. We follow the standard π-calculus interpretation for guarded choice, parallelism, restriction, and recursion (for which we assume $X \in \chi$). As usual, given $\sum_{i \in I} \alpha_i.P_i$, we write $\mathbf{0}$ when $|I| = 0$, and $\alpha_1.P_1 + \alpha_2.P_2$ when $|I| = 2$. We assume the usual definitions of free/bound variables and free/bound names for a process P, noted $fv(P), bv(P)$ and $fn(P), bn(P)$, respectively. The set of names of a process is defined as $n(P) = fn(P) \cup bn(P)$. Finally, notice that labels in $\mathcal{L}$ are not subject to restriction or binding.

The semantics of the CC is given as a labeled transition system (LTS). As customary, a transition $P \xrightarrow{\lambda} P'$ represents the evolution from P to P' through action λ. We write $P \xrightarrow{\lambda}$ if $P \xrightarrow{\lambda} P'$, for some P'. We define $P \rightarrow P'$ as $P \xrightarrow{\tau} P'$. We use $P \rightarrow^* P'$ to denote the transitive closure of $P \rightarrow P'$, and write $P \stackrel{\lambda}{\Rightarrow} P'$ when $P \rightarrow^* \xrightarrow{\lambda} \rightarrow^* P'$.

Definition 2. *Transition* labels λ *are defined in terms of* actions σ*, as defined by the following grammar:* $\sigma ::= \tau \mid \mathtt{l}^d?(\vec{x}) \mid \mathtt{l}^d!(\vec{n}) \mid \mathtt{this} \qquad \lambda ::= \sigma \mid c\,\sigma \mid (\nu n)\lambda$

Action τ denotes internal communication, while $\mathtt{l}^d?(\vec{x})$ and $\mathtt{l}^d!(\vec{n})$ represent an input and output to the environment, respectively. Action `this` represents a conversation identity access. A transition label λ can be either the (unlocated) action σ, an action σ *located at* conversation c (written $c\,\sigma$), or a transition label in which n is bound with scope λ. This is the case of bounded output actions. $out(\lambda)$ denotes the names produced by a transition, so $out(\lambda) = a$ if $\lambda = \mathtt{l}^d!(a)$ or $\lambda = c\mathtt{l}^d!(a)$ and $c \neq a$. A transition label λ denoting input/output communication, such as $\mathtt{l}^d?(\vec{x})$ or $\mathtt{l}^d!(\vec{n})$, is subject to *duality*, noted $\overline{\lambda}$. We write $\overline{\mathtt{l}^d?(\vec{x})} = \mathtt{l}^d!(\vec{n})$ and $\overline{\mathtt{l}^d!(\vec{n})} = \{\mathtt{l}^d?(\vec{x}) \mid \vec{x} \in \mathcal{V}\}$.

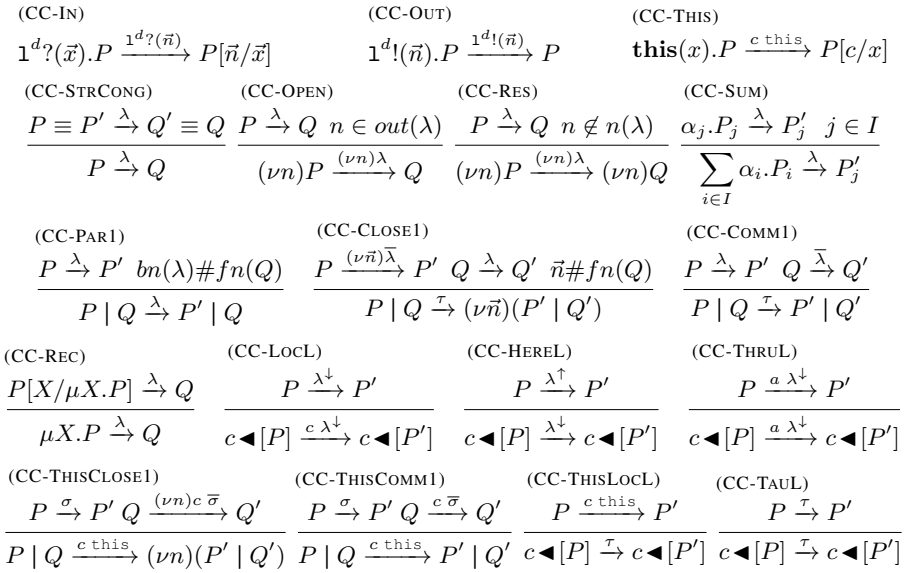

Fig. 3: An LTS for CC. Rules with labels ending with "1" have a symmetric counterpart (with label ending with "2") which is elided.

Fig. 3 presents the LTS. There, $\equiv$ stands for a structural congruence relation on processes; see [23] for details. The rules in the upper part of Fig. 3 follow the transition rules for a π-calculus with recursion. For instance, rule (CC-OPEN) corresponds to the usual scope extrusion rule. The rest of the rules are specific to the CC. Rule (CC-THIS) captures the name of an enclosing conversation context. Rule (CC-LOCL) locates an action to a particular conversation context, and rule (CC-HEREL) changes the direction of an action occurring inside a context. Rules (CC-THISCLOSE1) and (CC-THISCOMM1) are located versions of (CC-CLOSE) and (CC-COMM), respectively. Rule (CC-THISLOCL) hides an action occurring inside a conversation context. Rules (CC-THRUL) and (CC-TAUL) formalize how actions change when they "cross" a conversation context.

3 C3: CC with Time and Compensations

As anticipated in the Introduction, the syntax of C3 extends that of the CC with timed, compensable conversation contexts and a process for aborting running conversations:

Definition 3 (C3 **Syntax).** *The syntax of* C3 *processes is obtained from that given in Definition 1 by replacing the conversation contexts* $n \blacktriangleleft [P]$ *with* $n \blacktriangleleft [P\,;\, Q]^t_\kappa$ *(with* $n, \kappa \in \mathcal{N}$ *and* $t \in \mathbb{N}_0 \cup \{\infty\}$*) and by adding* $\kappa^\dagger$ *to the grammar of processes.*

Every notational convention introduced for CC processes carries over to C3 processes. In particular, as in the CC, notice that labels in $\mathcal{L}$ are not subject to restriction or binding. Unlike the LTS of CC, however, we assume a relation of structural congruence as in the π-calculus only (i.e., a congruence on processes with the usual axioms for α-conversion,

parallel composition, restriction, and $\mathbf{0}$). In particular, because of the timed nature of conversation contexts in C3 (on which we comment below), we refrain from adopting the axioms for manipulation of conversation contexts given in [23].

In C3, time is relative to each conversation context: it serves as a bound on the duration of the interactions *contained* in it. The time signature $t+1$ in a conversation context $c \blacktriangleleft [P\,;\, Q]^{t+1}_{\kappa}$ can evolve into t if the enclosing process P executes a "standard" action (i.e. any action except a compensation), or to 0 in case P fires a compensation. Hence, time in C3 is inherently *local* to each conversation context, rather than *global* to the whole system. We find this treatment of time in accordance with the intention of conversation contexts—distributed pieces of behavior in which a communication is organized. Put differently, since conversation contexts are essentially distributed abstractions of the participants of the multiparty interaction, considering a time signature local to each of them is a way of enforcing distribution. Also, as shown by our examples, this notion of time is convenient for the interplay with exceptional behavior.

The LTS for C3 is defined by the rules in Fig. 4; transition labels are obtained by extending the set of *actions* σ of the LTS of CC with a new action $\kappa^{\dagger}$, for an abortion (kill) process on κ. The convention on rule names for symmetric counterparts given in the LTS of the CC carries over to the LTS of C3. Moreover, for each of the left rules in Fig. 4—which describe evolution in the default behavior and have rule names ending in "L"—, there is an elided right rule characterizing evolution in the compensation behavior.

The passage of time in C3 is governed by the time elapsing function below. Intuitively, one time unit passes by as a consequence of the action. (Actions with durations different from one can be easily accommodated.)

Definition 4 (Time-elapsing function). *Given a* C3 *process P, we use $\phi(P)$ to denote the function that decreases the time bounds in P, inductively defined as:*

$$\phi(n \blacktriangleleft [Q\,;\, R]^{t+1}_{\kappa}) = n \blacktriangleleft [\phi(Q)\,;\, R]^{t}_{\kappa} \quad \phi(P \mid Q) = \phi(P) \mid \phi(Q) \quad \phi((\nu n)P) = (\nu n)\phi(P)$$
$$\phi(n \blacktriangleleft [Q\,;\, R]^{0}_{\kappa}) = n \blacktriangleleft [Q\,;\, \phi(R)]^{0}_{\kappa} \qquad \phi(P) = P \quad \textit{Otherwise.}$$

Given $k > 0$, we define $\phi^k(P) = \phi(P)$ if $k = 1$ and $\phi^k(P) = \phi(\phi^{k-1}(P))$, otherwise.

We describe some representative rules in Fig. 4. Rule (PAR1) decrees that executing an action in P decreases in one the time signatures of the conversation contexts in Q. This is the only rule that appeals to the time-elapsing function. For example, assuming a process P such that $P \xrightarrow{\lambda} P'$, using rules (LOCL), (RES), and (PAR1), we can infer that process $(\nu n)(c \blacktriangleleft [P\,;\, Q]^{t_1}_{\kappa_1}) \mid c \blacktriangleleft [R\,;\, S]^{t_2}_{\kappa_2}$ performs λ and evolves into $(\nu n)(c \blacktriangleleft [P'\,;\, Q]^{t_1-1}_{\kappa_1}) \mid \phi(c \blacktriangleleft [R\,;\, S]^{t_2}_{\kappa_2})$. Rules formalizing communication and closing of scope extrusion do not affect the passage of time. In process $e \blacktriangleleft [d \blacktriangleleft [(\nu n)(c \blacktriangleleft [P\,;\, Q]^{t_1}_{\kappa_1}) \mid c \blacktriangleleft [R\,;\, S]^{t_2}_{\kappa_2}\,;\, U]^{t_3}_{\kappa_3}\,;\, V]^{t_4}_{\kappa_4}$, timer t_4 will be updated after applications of (THRUL), unless a transition $c \blacktriangleleft [R\,;\, S]^{t_2}_{\kappa_2} \xrightarrow{\bar{\lambda}}$ can be inferred, in which case rules (THISCLOSE1) and (THISLOCL) are applied and timers t_1, t_2, t_3 are updated. This means that only the actions leading to a synchronization—but not the synchronization itself—contribute to the passage of time. Visible actions are privileged in the sense that they affect the time bound of the enclosing conversation context—compare rules (THRUL) and (TAUL).

(IN) $\mathtt{1}^d?(\vec{x}).P \xrightarrow{\mathtt{1}^d?(\vec{n})} P[\vec{n}/\vec{x}]$

(OUT) $\mathtt{1}^d!(\vec{n}).P \xrightarrow{\mathtt{1}^d!(\vec{n})} P$

(THIS) $\mathbf{this}(x).P \xrightarrow{c\ \mathtt{this}} P[c/x]$

(OPEN) $$\frac{P \xrightarrow{\lambda} Q \quad n \in out(\lambda)}{(\nu n)P \xrightarrow{(\nu n)\lambda} Q}$$

(RES) $$\frac{P \xrightarrow{\lambda} Q \ \ n \notin n(\lambda)}{(\nu n)P \xrightarrow{(\nu n)\lambda} (\nu n)Q}$$

(SUM) $$\frac{\alpha_j.P_j \xrightarrow{\lambda} P'_j \ \ j \in I}{\sum_{i\in I} \alpha_i.P_i \xrightarrow{\lambda} P'_j}$$

(CLOSE1) $$\frac{P \xrightarrow{(\nu\vec{n})\overline{\lambda}} P' \ \ Q \xrightarrow{\lambda} Q' \ \ \vec{n}\#fn(Q)}{P \mid Q \xrightarrow{\tau} (\nu\vec{n})(P' \mid Q')}$$

(COMM1) $$\frac{P \xrightarrow{\lambda} P' \ \ Q \xrightarrow{\overline{\lambda}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

(REC) $$\frac{P[X/\mu X.P] \xrightarrow{\lambda} Q}{\mu X.P \xrightarrow{\lambda} Q}$$

(PAR1) $$\frac{P \xrightarrow{\lambda} P' \ \ bn(\lambda)\#fn(Q)}{P \mid Q \xrightarrow{\lambda} P' \mid \phi(Q)}$$

(THISCOMM1) $$\frac{P \xrightarrow{\sigma} P' \quad Q \xrightarrow{c\ \overline{\sigma}} Q'}{P \mid Q \xrightarrow{c\ \mathtt{this}} P' \mid Q'}$$

(THISCLOSE1) $$\frac{P \xrightarrow{\sigma} P' \quad Q \xrightarrow{(\nu n)c\ \overline{\sigma}} Q'}{P \mid Q \xrightarrow{c\ \mathtt{this}} (\nu n)(P' \mid Q')}$$

(TAUL) $$\frac{P \xrightarrow{\tau} P'}{c\blacktriangleleft[P\,;\,Q]^t_\kappa \xrightarrow{\tau} c\blacktriangleleft[P'\,;\,Q]^t_\kappa}$$

(THISLOCL) $$\frac{P \xrightarrow{c\ \mathtt{this}} P'}{c\blacktriangleleft[P\,;\,Q]^{t+1}_\kappa \xrightarrow{\tau} c\blacktriangleleft[P'\,;\,Q]^t_\kappa}$$

(THRUL) $$\frac{P \xrightarrow{a\ \lambda^\downarrow} P'}{c\blacktriangleleft[P\,;\,Q]^{t+1}_\kappa \xrightarrow{a\ \lambda^\downarrow} c\blacktriangleleft[P'\,;\,Q]^t_\kappa}$$

(LOCL) $$\frac{P \xrightarrow{\lambda^\downarrow} P'}{c\blacktriangleleft[P\,;\,Q]^{t+1}_\kappa \xrightarrow{c\ \lambda^\downarrow} c\blacktriangleleft[P'\,;\,Q]^t_\kappa}$$

(HERELL) $$\frac{P \xrightarrow{\lambda^\uparrow} P'}{c\blacktriangleleft[P\,;\,Q]^{t+1}_\kappa \xrightarrow{\lambda^\downarrow} c\blacktriangleleft[P'\,;\,Q]^t_\kappa}$$

(ABORT) $\kappa^\dagger \xrightarrow{\kappa^\dagger} \mathbf{0}$

(FAILPAR1) $$\frac{P \xrightarrow{\kappa^\dagger} P'}{P \mid c\blacktriangleleft[Q\,;\,R]^t_\kappa \xrightarrow{\tau} P' \mid c\blacktriangleleft[Q\,;\,R]^0_\kappa}$$

(COMP) $$\frac{Q \xrightarrow{\lambda} Q'}{c\blacktriangleleft[P\,;\,Q]^0_\kappa \xrightarrow{\lambda} c\blacktriangleleft[P\,;\,Q']^0_\kappa}$$

(FAILTHRUL) $$\frac{P \xrightarrow{\kappa^\dagger} P' \qquad \kappa \neq \gamma}{c\blacktriangleleft[P\,;\,Q]^t_\gamma \xrightarrow{\kappa^\dagger} c\blacktriangleleft[P'\,;\,Q]^t_\gamma}$$

(FAILINT) $$\frac{P \xrightarrow{\kappa^\dagger} P'}{c\blacktriangleleft[P\,;\,Q]^t_\kappa \xrightarrow{\tau} c\blacktriangleleft[P'\,;\,Q]^0_\kappa}$$

Fig. 4: Rules for the LTS of C3.

Rules (ABORT), (FAILPAR1), and (COMP) handle abortion signals and exceptional behavior in C3: the first formalize such signals; the second represents the abortion of a conversation context; the third formalizes the behavior of an aborted conversation context. Informally, compensation signals travel vertically upwards over levels of nested conversation contexts: following (FAILTHRUL) in process $c \blacktriangleleft [d \blacktriangleleft [\kappa_c^\dagger\,;\,Q]^{t_d}_{\kappa_d}\,;\,R]^{t_c}_{\kappa_c}$, action $\kappa_c^\dagger$ will cross d so as to affect c, its the outermost conversation context. Compensation signals can also affect surrounding conversation contexts: using (FAILPAR1) in process $c\blacktriangleleft[\kappa_d^\dagger\,;\,Q]^{t_c}_{\kappa_c} \mid d\blacktriangleleft[R\,;\,S]^{t_d}_{\kappa_d}$, will activate the alternative behavior in conversation context d (assuming $\kappa_c \neq \kappa_d$). Finally, compensation signals become unobservable if they lead to abortion (cf. rule (FAILINT)).

$$\text{(TC1)}\quad \texttt{throw}.P \xrightarrow{\texttt{throw}} P \qquad \text{(TC2)}\quad \frac{P \xrightarrow{\texttt{throw}} R}{P \mid Q \xrightarrow{\texttt{throw}} R} \qquad \text{(TC3)}\quad \frac{P \xrightarrow{\texttt{throw}} R}{n \blacktriangleleft [P] \xrightarrow{\texttt{throw}} R}$$

$$\text{(TC4)}\quad \frac{P \xrightarrow{\lambda} P' \quad \lambda \neq \texttt{throw}}{\texttt{try}\, P\, \texttt{catch}\, Q \xrightarrow{\lambda} \texttt{try}\, P'\, \texttt{catch}\, Q} \qquad \text{(TC5)}\quad \frac{P \xrightarrow{\texttt{throw}} R}{\texttt{try}\, P\, \texttt{catch}\, Q \xrightarrow{\tau} Q} \qquad \text{(TC6)}\quad \frac{P \xrightarrow{\texttt{throw}} R}{\texttt{try}\, P\, \texttt{catch}\, Q \xrightarrow{\tau} Q \mid R}$$

Fig. 5: LTS rules for an extension of the CC with try-catch

4 The Expressiveness of C3, Informally

With the purpose of illustrating some of its features, here we compare C3 with two of its sublanguages. The first sublanguage, noted $\mathsf{C3}^{-k}$, is C3 without compensation signals, while the second, noted $\mathsf{C3}^{-t}$, corresponds to C3 without time. Hence, while in $\mathsf{C3}^{-k}$ there are no explicit abortion signals and conversation contexts have time bounds, in $\mathsf{C3}^{-t}$ there are abortion signals, but all conversation contexts have ∞ as time bound. The semantics of each fragment can be obtained from that for C3 with the expected modifications. Our treatment is largely informal, as our objective is to shed light on the nature of C3. Formal comparisons of relative expressiveness are left for future work.

C3 **and constructs for exception handling.** We consider the extension of the CC given in Section 2 with a try-catch operator. So we extend the syntax in Definition 1 with a new process construct $\texttt{try}\, P\, \texttt{catch}\, Q$ and a new action `throw`. The LTS of CC is extended with rules TC1-TC5 in Fig. 5. These rules give the semantics of exception handling considered in works such as, e.g., [3]. Let us refer to this extension as $\mathsf{CC}^{\mathsf{tc1}}$.

In $\mathsf{C3}^{-t}$ we can model a try-catch construct with such a semantics. Consider an encoding (language translation) $[\![\cdot]\!]^{tc}$, an homomorphism for every operator except for:

$$[\![\texttt{try}\, P\, \texttt{catch}\, Q]\!]^{tc} = n \blacktriangleleft [[\![P]\!]^{tc}\,;\, [\![Q]\!]^{tc}]^{\infty}_{\kappa_n} \qquad [\![\texttt{throw}]\!]^{tc} = \kappa^{\dagger}_n$$

where κ_n and n are distinguished fresh names. The encoding captures the semantics of the try-catch operator thanks to the fact that in C3 a compensation signal (a process $\kappa^{\dagger}_n$) can have effect on the conversation context enclosing it (cf. rule FAILINT in Fig. 4).

A compositional encoding in the opposite direction, i.e., an encoding of $\mathsf{C3}^{-t}$ into $\mathsf{CC}^{\mathsf{tc1}}$, in which try-catch blocks are used to model conversation contexts, appears difficult. This is because of the nature of compensation in C3 (and $\mathsf{C3}^{-t}$): extended conversation contexts can be aborted by both *internal* and *external* signals (cf. rules (FAILINT) and (FAILPAR1) in Fig. 4), and not only by signals inside the try block. A possibility is to define an encoding that, using the number of conversation contexts and abortion signals in the process, creates a try-catch for every possible combination. This is a rather unsatisfactory solution, as interactions may give rise to new conversation contexts, and so the number of "global" try-catch constructs required for the encoding may not be predictable in advance. Based on these observations, one could conjecture that $\mathsf{C3}^{-t}$ is strictly more expressive than $\mathsf{CC}^{\mathsf{tc1}}$.

Notice that an encoding such as the above would not work with a CC with try-catch with a different semantics. For instance, semantics enforcing advanced treatment of nested try blocks [26] may be difficult to handle. Let us consider $\mathsf{CC}^{\mathsf{tc2}}$, the extension

of CC with the semantics given by rules TC1-TC4 and TC6 in Fig. 5. This is the semantics used in, e.g., [4]. The crucial difference between rules TC5 and TC6 is that in the latter the state of the try-block just after the exception has been raised (i.e., R in both rules) is preserved when the exception block (i.e., Q) is called for execution, while in the former such a state is discarded. It is not obvious how to represent such a preservation of state in C3, as the standard part of the conversation context is completely discarded when the context is aborted, and there is no way of accessing it afterwards. We thus conjecture the non existence of an encoding of $\mathsf{CC}^{\mathsf{tc2}}$ into $\mathsf{C3}^{-t}$.

Time and Interruptions in C3. From the point of view of exceptional behavior, time in C3 can be assimilated to an interruption mechanism over the standard part of a conversation context. We now informally claim that this character of timed behavior represents a difference in expressive power between $\mathsf{C3}^{-k}$ and the extensions of CC with try-catch described above.

Let us first consider $\mathsf{CC}^{\mathsf{tc1}}$. An encoding of $\mathsf{C3}^{-k}$ into $\mathsf{CC}^{\mathsf{tc1}}$ could exploit the fact that the semantics of try-catch allows to interrupt the behavior of a process placed in the try block. This is true even for persistent processes, such as $P = \mu X.\mathtt{l}_1!(n).X$. However, it is not obvious at all where to place the `throw` prefixes so as to properly model the passage of time. That is, process interruption could be encoded but not at the *right time*: hence, it is not possible to guarantee that the behavior of the $\mathsf{C3}^{-k}$ process is faithfully captured. Therefore, we conjecture that there is no encoding of $\mathsf{C3}^{-k}$ into $\mathsf{CC}^{\mathsf{tc1}}$, up to some notion of operational correspondence sensible to timed behavior. When considering $\mathsf{CC}^{\mathsf{tc2}}$, the conjecture appears somewhat more certain since, as discussed before, the semantics of $\mathsf{CC}^{\mathsf{tc2}}$ does preserve the last state of the try block before the exception is raised. That is, such a semantics does not implement interruption of the try block. This way, using P defined as above, process $S = \mathtt{try}\, P \parallel \mathtt{throw}.\mathbf{0}\, \mathtt{catch}\, \mathtt{l}_2!(n)$ would exhibit persistent behavior, even after the exception has been raised.

5 A Healthcare Compelling Example

Here we present a compelling example for C3: a medicine delivery scenario, adapted from [18,6], which features time and exceptional behavior. After presenting the scenario, we present two models: while the first one is a basic model in CC, the second one is a model in C3 which allows a more comprehensive specification of the scenario.

The Medicine Delivery Scenario. Alice is a patient recently discharged from a hospital after a cardiac arrest. Sensors attached to Alice monitor her health conditions 24 hours a day; data is controlled by an Emergency Response Center (EC). The medicine delivery scenario takes place when Alice feels weak and, instead of driving to the pharmacy to get the medicine, asks to be supported by the EC. To this end, the EC requests a Social Worker (SW) to bring the medicine to Alice. There are both *mobile* SWs (dealing with requests outside the EC) and *in-house* SWs (the rest). If none of the mobile SWs can attend the request then an in-house SW is contacted. The selected SW gets appointed by the EC by sending him authorization keys for receiving the medicine and communicating with Alice. The SW can now acknowledge the request and go to the

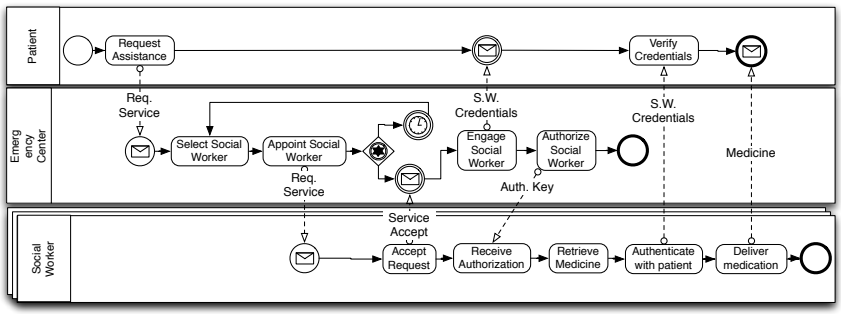

Fig. 6: BPMN diagram for the medicine delivery scenario.

pharmacy. After a successful message exchange between Alice's terminal and the EC, the SW is authenticated and entitled to receive the medicine. Finally, the SW must authenticate to Alice in order to deliver the medicine. Fig. 6 gives a specification of the scenario in the Business Process Modeling Notation (BPMN 2.0), the de-facto notation for the specification of executable business processes.

In addition to the expected functional requirements, non-functional requirements related to the *availability* of the system are also important in the medicine delivery scenario. For instance, it is necessary to ensure that in the exceptional circumstance in which there are no available social workers nearby the patient, there is always someone who can complete the process and provide the medication. These are local policies (as they rule the execution of an agent in the system) but have effects at a global level (as they may refer to other agents apart from the ones currently executing). Timed aspects intrinsically related to the scenario are also inherently global. For instance, studies in [22] show that patients with out-of-hospital cardiac arrests who are not provided with medicine within 17 minutes have a higher chance of death. This kind of policies do not pertain the behavior of a single agent, but they involve global conditions applied to the whole interaction of roles involved in the process. Thus, they are global policies which have specific consequences in the involved principals.

The Scenario in CC **and** C3**.** Our first model of the scenario is a CC specification, given as process T in Fig. 7. The model features three interacting agents, defined as processes $Patient$, EC, and SW. Each one defines a conversation context (noted n, e, and s, resp.). Interactions occur between $Patient$ and EC first, then between EC and SW, and finally between SW and $Patient$. More precisely, $Patient$ starts the protocol by invoking service AB, located at e. The body of $\mathbf{new}\, e \cdot AB \Leftarrow \cdots$ first receives a request from $Patient$, and then extends the established conversation so as to include SW, using the idiom $\mathbf{join}\, s \cdot BC \Leftarrow \cdots$. Once $Patient$, EC, and SW share the conversation, they are able to interact. This is evident in, e.g., interactions on `c` and `acc` between EC and SW and interactions on `a` between $Patient$ and EC.

While the CC specification is useful to describe the basic interacting behavior in the medicine delivery scenario, additional elements are required in order to completely faithful with the functional and non-functional requirements of the scenario. Some of

$$T \triangleq Patient \mid EC \mid SW \quad \text{where:}$$

$$Patient \triangleq n \blacktriangleleft [\mathbf{new}\, e \cdot AB \Leftarrow \mathtt{b}!(l).\mathtt{a}?(z,m).\mathtt{a2}?(z,m).\mathtt{a3}?(p)]$$

$$EC \triangleq e \blacktriangleleft [\star\mathbf{def}\, AB \Rightarrow \mathtt{b}?(y).\mathbf{join}\, s \cdot BC \Leftarrow \mathtt{c}!(y).\mathtt{acc}?(v).(\nu pk)\mathtt{a}!(v,pk).\mathtt{c}!(pk)]$$

$$SW \triangleq s \blacktriangleleft [\mathbf{def}\, BC \Rightarrow \mathtt{c}?(z).((\nu v)\mathtt{acc}!(v).\mathtt{c}?(m_n).\mathtt{a2}!(n,m_n).\mathtt{a3}!(p))]$$

Fig. 7: Medicine Delivery Scenario: Basic Model in CC

such elements are difficult to express in CC. The specification in C3, given as process S in Fig. 8, allows to give a more comprehensive account of the scenario.

In the extended model, we consider a new agent, $Nurse$, representing a nurse and defining a conversation context u. The interaction patterns follow the lines of the previous model, with a few extensions. For instance, the body of the service AB located at e contains calls to DB, a database containing the contacts of available SWs. The direction of the messages between DB and the service is $\uparrow$, as communications have to cross the boundary defined by the service definition. AB describes the process that first selects one of the SWs and then forwards the request for attention started by the patient. Process SW_i represents the behavior of the i-th available mobile SW. Following the scenario (and unlike the model in CC), in the C3 specification the SW can either accept or ignore the request; in the latter case the process iterates until some SW is appointed. Upon acceptance, EC will generate new credentials identifying the SW (represented in the model with a fresh name m); in turn, these will be transmitted to the patient for further checking. Besides this extended behavior with respect to SWs, two important exceptional behaviors can be observed in this example. The first one concerns the prompt response required by the patient. In case the request for attention is not delivered on due time and the patient starts feeling dizzy, sensors attached to patient's body can detect problems in her health conditions (say, low blood pressure) and restart automatically the medicine delivery process in order to ensure the request call is answered. The behavior of the sensors here is abstracted by a non-deterministic choice; more detailed specifications are of course possible. This behavior is present in the alternative behavior for conversation context n, and it will be fired either when the expected time t_A has passed by, or when EC reports unavailability (represented by process $\kappa_A^\dagger$). The second timeout refers to internal process requirements from EC, which stipulate that each request for attention has to be attended within t_B time units. This is irrespective from any further communication done elsewhere. The specification for the EC is thus fault tolerant, and on unavailability of a mobile SW, it will rely on service BD offered by $Nurse$.

6 Related Work

Although there is a long history of timed extensions for process calculi (see, e.g., [1]) and the study of constructs for exceptional behavior has received significant attention (see [11] for an overview), time and its interplay with forms of exceptional behavior do not seem to have been jointly studied in languages for structured communication.

$S \triangleq Patient \mid EC \mid SW_i \mid Nurse$ where

$Patient \triangleq n \blacktriangleleft [\mathbf{new}\, e \cdot AB \Leftarrow \mathtt{b}!(l).(\mathtt{a}?(z,m).\mathtt{a2}?(z,m).\mathtt{a3}?(p) + \kappa_A^\dagger)\,;$
$\mathbf{new}\, e \cdot AB \Leftarrow \mathtt{b}!(l).\mathtt{a}?(z,m).\mathtt{a2}?(z,m).\mathtt{a3}?(p)]_{\kappa_A}^{t_A}$

$EC \triangleq e \blacktriangleleft [DB \mid \star\, \mathbf{def}\, AB \Rightarrow \mathtt{b}?(y).\mu X.\mathtt{req}^\uparrow!(y).\mathtt{rep}^\uparrow?(n).$
$\mathbf{join}\, s_i \cdot BC \Leftarrow \mathtt{c}!(y).(\mathtt{dny}?(v).X + \mathtt{acc}?(v).(\nu m)\mathtt{a}!(v,m).\mathtt{c}!(m))\,;$
$\kappa_A^\dagger \mid DB \mid \star\, \mathbf{def}\, AB \Rightarrow \mathtt{b}?(y).\mathbf{join}\, u \cdot BD \Leftarrow \mathtt{d}!(y).\mathtt{acc}?(v).(\nu m)\mathtt{a}!(v,m).\mathtt{d}!(m)]_{\kappa_B}^{t_B}$

$SW_i \triangleq s_i \blacktriangleleft [\mathbf{def}\, BC \Rightarrow \mathtt{c}?(z).((\nu v)(\mathtt{dny}!(v) + \mathtt{acc}!(v)).\mathtt{c}?(m_n).\mathtt{a2}!(n,m_n).\mathtt{a3}!(p))\,;\, \mathbf{0}]_{\kappa_C}^{t_C}$

$Nurse \triangleq u \blacktriangleleft [\mathbf{def}\, BD \Rightarrow \mathtt{d}?(z).((\nu v)\mathtt{acc}!(v).\mathtt{d}?(m).\mathtt{a2}!(k,m).\mathtt{a3}!(p))\,;\, \mathbf{0}]_{\kappa_D}^{\infty}$

Fig. 8: The medicine delivery scenario in C3

Our previous work [19] reported an LTL interpretation of the session language in [12] and proposed extensions with time, declarative information, and a construct for session abortion. The language in [12], however, does not support multiparty interactions. The differences in expressiveness between C3 and a variant of the CC featuring try-catch constructs [24] have been already discussed in Section 4.

Time and exceptional behavior have been considered only separately in orchestrations and choreographies. As for time, Timed Orc [25] introduced real-time observations for orchestrations by introducing a delay operator, while Timed COWS [16] extends COWS (the Calculus for Orchestration of Web Services [15]) with operators of delimitation, abortion, and delays for orchestrations. As for exceptional behavior, [10,7] propose languages for *interactional exceptions*, in which exceptions in a protocol generate coordinated actions between all peers involved. Associated type systems ensure communication safety and termination among protocols with normal and compensating executions. In [7], the language is enriched further with multiparty session and global escape primitives, allowing nested exceptions to occur at any point in an orchestration. As for choreographies, [8] introduced an extension of a language of choreographies with try/catch blocks, guaranteeing that embedded compensating parts in a choreography are not arbitrarily killed as a result of an abortion signal. Similarly, [27] presents a semantics for language for choreographies with exception handling and finalization constructs, which allows a projection from exceptional behavior of a choreography to its endpoints. In comparison to C3, such a semantics enforces a different (centralized) treatment of choice, defines compensation blocks as exceptions in sequential languages, and does not provide support for nested compensating blocks.

Our work has been influenced by extensions to the (asynchronous) π-calculus, notably [14,1]. In particular, the rôle of the time-elapsing behavior for conversation contexts used in C3 draws inspiration from the behavior of long transactions in webπ [14], and from the π-calculus with timers in [1]. The nature of the languages in [14,1] and C3 is very different. In fact, while C3 is a synchronous language, the calculi in [14,1] are asynchronous. Also, webπ is a language for study long-running transactions; hence, exceptions in webπ and compensations in C3 have different meanings, even the constructs appear syntactically similar.

7 Concluding Remarks

We have reported initial steps towards a joint study of time and exceptional behavior in the context of multiparty structured communications. We have presented C3, a variant of the CC in which conversation contexts have an explicit duration, a compensation activity, and can be explicitly aborted. The expressiveness and relevance of its two main features (explicit abortion signals and timed behavior) have been illustrated in a compelling example extracted from a healthcare scenario of structured communications.

There are a number of directions which are worth pursuing based on the developments presented here; we briefly mention some of them. The most pressing issue concerns analysis techniques for C3 specifications. We would like to develop type disciplines for ensuring communication correctness in models featuring time and exceptional behavior. For this purpose, conversation types [5] and the linear/affine type system proposed in [2] might provide a reasonable starting point. We are also interested in a notion of *refinement* between a model in CC and an associated model in C3. In [17, Chapter 7] we report some preliminary ideas in this direction: intuitively, the objective is to decree that a C3 model is a refinement of a related CC model if they pass a set of *tests* present in both models; the challenge is to obtain suitable characterizations for such tests, considering time and the execution of compensating behavior. Finally, we would like to obtain separation results for the expressiveness conjectures stated in Section 4. As we have pointed out, different models for exception handling induce different semantics for treating exceptional behavior; it would be interesting to understand their precise relation in terms of expressiveness. While some previous work has addressed similar questions [13], we think it would be relevant to carry out similar studies in the context of languages for structured communications, such as CC and C3.

Acknowledgments. This research has been supported by the Danish Research Agency through the Trustworthy Pervasive Healthcare Services project (grant #2106-07-0019, www.TrustCare.eu) and by the Portuguese Foundation for Science and Technology (FCT/MCTES) through the Carnegie Mellon Portugal Program, grant INTERFACES NGN-44 / 2009. We thank the anonymous reviewers for their useful comments.

References

1. M. Berger and K. Honda. The two-phase commitment protocol in an extended pi-calculus. *Electr. Notes Theor. Comput. Sci.*, 39(1), 2000.
2. M. Berger and N. Yoshida. Timed, distributed, probabilistic, typed processes. In *Proc. APLAS*, volume 4807 of *LNCS*, pages 158–174. Springer, 2007.
3. M. Bravetti and G. Zavattaro. On the expressive power of process interruption and compensation. *Mathematical Structures in Computer Science*, 19(3):565–599, 2009.
4. L. Caires, C. Ferreira, and H. T. Vieira. A process calculus analysis of compensations. In *Proc. of TGC'08*, volume 5474 of *LNCS*, pages 87–103. Springer, 2008.
5. L. Caires and H. T. Vieira. Conversation types. *Theor. Comput. Sci.*, 411(51-52):4399–4440, 2010.
6. S. Campadello, L. Compagna, D. Gidoin, S. Holtmanns, V. Meduri, J. Pazzaglia, M. Seguran, and R. Thomas. Scenario Selection and Definition. Research report A7.D1.1, SERENITY consortium, 2006.

7. S. Capecchi, E. Giachino, and N. Yoshida. Global Escape in Multiparty Sessions. In K. Lodaya and M. Mahajan, editors, *FSTTCS'2010*, volume 8 of *LIPIcs*, pages 338–351, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
8. M. Carbone. Session-based choreography with exceptions. In *PLACES'08*, volume 241 of *ENTCS*, pages 35–55, 2008.
9. M. Carbone, K. Honda, and N. Yoshida. Structured communication-centred programming for web services. In *Proc. of ESOP'07*, volume 4421 of *LNCS*, pages 2–17. Springer, March 2007.
10. M. Carbone, K. Honda, and N. Yoshida. Structured Interactional Exceptions in Session Types. In *CONCUR'2008*, LNCS, pages 402–417. Springer-Verlag, 2008.
11. C. Ferreira, I. Lanese, A. Ravara, H. T. Vieira, and G. Zavattaro. Advanced mechanisms for service combination and transactions. In *Rigorous Software Engineering for Service-Oriented Systems - Results of the SENSORIA project*, volume 6582 of *LNCS*. Springer, 2011.
12. K. Honda, V. Vasconcelos, and M. Kubo. Language Primitives and Type Discipline for Structured Communication-Based Programming. In *Proc. of ESOP'98*, pages 122–138. Springer-Verlag London, UK, 1998.
13. I. Lanese, C. Vaz, and C. Ferreira. On the expressive power of primitives for compensation handling. In *Proc. of ESOP*, volume 6012 of *LNCS*, pages 366–386. Springer, 2010.
14. C. Laneve and G. Zavattaro. Foundations of web transactions. In *Proc. of FoSSaCS'05*, volume 3441 of *LNCS*, pages 282–298. Springer, 2005.
15. A. Lapadula, R. Pugliese, and F. Tiezzi. A calculus for orchestration of web services. In *ESOP'07*, volume 4421 of *LNCS*, pages 33–47. Springer, 2007.
16. A. Lapadula, R. Pugliese, and F. Tiezzi. C-clock-WS: a timed service-oriented calculus. In *Proc. of ICTAC'07*, pages 275–290. Springer-Verlag, 2007.
17. H. A. López. *Foundations of Communication-Centred Programming*. PhD thesis, IT University of Copenhagen, 2012.
18. H. A. López, F. Massacci, and N. Zannone. Goal-Equivalent Secure Business Process Reengineering. In *ICSOC 2007 workshops*, volume 4907 of *LNCS*, pages 212—223, Berlin, Heidelberg, January 2009. Springer.
19. H. A. López, C. Olarte, and J. A. Pérez. Towards a unified framework for declarative structured communications. In *PLACES*, volume 17 of *EPTCS*, pages 1–15, 2009.
20. K. Lyng, T. Hildebrandt, and R. Mukkamala. From paper based clinical practice guidelines to declarative workflow management. In *Business Process Management Workshops*, pages 336–347. Springer, 2009.
21. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts I and II. *Journal of Information and Computation*, 100:1–77, September 1992.
22. J. C. Rittenberger, J. E. Bost, and J. J. Menegazzi. Time to give the first medication during resuscitation in out-of-hospital cardiac arrest. *Resuscitation*, 70(2):201–6, Aug 2006.
23. H. T. Vieira. *A Calculus for Modeling and Analyzing Conversations in Service-Oriented Computing*. PhD thesis, Universidade Nova de Lisboa, 2010.
24. H. T. Vieira, L. Caires, and J. C. Seco. The conversation calculus: A model of service-oriented computation. In *Proc. of ESOP'08*, volume 4960 of *LNCS*, pages 269–283. Springer, 2008.
25. I. Wehrman, D. Kitchin, W. R. Cook, and J. Misra. A timed semantics of orc. *Theor. Comput. Sci.*, 402(2-3):234–248, 2008.
26. J. Xu, A. B. Romanovsky, and B. Randell. Coordinated exception handling in distributed object systems: From model to system implementation. In *ICDCS*, pages 12–21, 1998.
27. H. Yang, X. Zhao, C. Cai, and Z. Qiu. Exploring the connection of choreography and orchestration with exception handling and finalization/compensation. In *Proc. of FORTE'2007*, volume 4574 of *LNCS*, pages 81–96. Springer, 2007.